CHAPTER 3

Design for testability

Laung-Terng (L.-T.) Wang
SynTest Technologies, Inc., Sunnyvale, California

ABOUT THIS CHAPTER

Design for testability (DFT) has become an essential part for designing ***very-large-scale integration*** (VLSI) circuits. The most popular DFT techniques in use today for testing the digital portion of the VLSI circuits include **scan** and **scan-based logic *built-in self-test*** (BIST). Both techniques have proved to be quite effective in producing testable VLSI designs. In addition, **test compression**, a supplemental DFT technique for scan, is growing in importance for further reduction in test data volume and test application time during manufacturing test.

To provide readers with an in-depth understanding of the most recent DFT advances in scan, logic BIST, and test compression, this chapter covers a number of fundamental DFT techniques to facilitate testing of modern digital circuits. These techniques are required to improve the product quality and reduce the defect level and test cost of a digital circuit, while at the same time simplifying the test, debug, and diagnosis tasks.

In this chapter, we first cover the basic DFT concepts and methods for performing testability analysis. Next, **scan design**, the most widely used structured DFT method, is discussed, including popular scan cell designs, scan architectures, and at-speed clocking schemes. After a brief introduction to the basic concept of logic BIST, we then discuss BIST pattern generation and output response analysis schemes along with a number of logic BIST architectures for in-circuit self-test. Finally, we present a number of test compression circuit structures for test stimuli compression and test response compaction. The chapter also includes a description of logic BIST and test compression architectures currently practiced in industry.

3.1 INTRODUCTION

With advances in semiconductor manufacturing technology, ***integrated circuits*** (ICs) can now contain tens to hundreds of millions of transistors running in the gigahertz range. The production and use of these integrated circuits has run into a variety of test challenges during wafer probe, wafer sort, preship screening, incoming test of chips and boards, test of assembled boards, system test, periodic maintenance, repair test, etc. During the early stages of IC production history, design and test were regarded as separate functions, performed by separate and unrelated groups of engineers. During these early years, a design engineer's job was to implement the required functionality on the basis of design specifications, without giving any thought to how the manufactured device was to be tested. Once the functionality was implemented, the design information was transferred to test engineers. A test engineer's job was to determine how to best test each manufactured device within a reasonable amount of time and to screen out the parts that may contain manufacturing defects while shipping all defect-free devices to customers. The final quality of the test was determined by keeping track of the number of defective parts shipped to the customers on the basis of customer returns. This product quality, measured in terms of ***defective parts per million*** (DPM) shipped, was a final test score for quantifying the effectiveness of the developed test.

Although this approach worked well for small-scale integrated circuits that mainly consisted of combinational logic or simple finite-state machines, it was unable to keep up with the circuit complexity as designs moved from ***small-scale integration*** (SSI) to ***very large-scale integration*** (VLSI). A common approach to testing these VLSI devices during the 1980s relied heavily on fault simulation to measure the fault coverage of the supplied functional patterns. Functional patterns were developed to navigate through the long sequential depths of a design, hoping to exercise all internal states and to detect all possible manufacturing defects. A **fault simulation** or **fault-grading** tool was used to quantify the effectiveness of the functional patterns. If the supplied functional patterns did not reach the target fault coverage goal, additional functional patterns were added. Unfortunately, this approach typically failed to improve the circuit's fault coverage beyond 80%, and the quality of the shipped products suffered.

Gradually, it became clear that designing devices without paying much attention to test resulted in increased test cost and decreased test quality. Some designs, which were otherwise best-in-class with regard to functionality and performance, failed commercially because of prohibitive test costs or poor product quality. These problems have since led to the development and deployment of DFT engineering in the industry.

The first challenge facing DFT engineers was to find simpler ways of exercising all internal states of a design and reaching the target fault coverage goal.

Various **testability measures** and **ad hoc testability enhancement** methods were proposed and used in the 1970s and 1980s to serve this purpose. These methods were mainly used to aid in the circuit's **testability** or to increase the circuit's **controllability** and **observability** [McCluskey 1986; Abramovici 1994]. Although attempts to use these methods have substantially improved the testability of a design and eased sequential ***automatic test pattern generation*** (ATPG), their end results at reaching the target fault coverage goal were far from satisfactory; it was still quite difficult to reach more than 90% fault coverage for large designs. This was mostly because even with these testability aids, deriving functional patterns by hand or generating test patterns for a sequential circuit is a much more difficult problem than generating test patterns for a combinational circuit [Fujiwara 1982; Bushnell 2000; Jha 2003].

Today, the semiconductor industry relies heavily on two techniques for testing digital circuits: *scan* and *logic built-in self-test* (BIST) [Abramovici 1994; McCluskey 1986]. **Scan** converts a digital sequential circuit into a scan design and then uses ATPG software [Bushnell 2000; Jha 2003; Wang 2006a] to detect faults that are caused by manufacturing defects (physical failures) and manifest themselves as errors, whereas logic BIST requires the use of a portion of the VLSI circuit to test itself on-chip, on-board, or in-system. To keep up with the design and test challenges [SIA 2005, 2006], more advanced ***design-for-testability*** (DFT) techniques, such as test compression, at-speed delay fault testing, and power-aware test generation, have been developed over the past few years to further address the test cost, delay fault, and test power issues [Gizopoulos 2006; Wang 2006a, 2007a].

Scan design is implemented by first replacing all selected storage elements of the digital circuit with **scan cells** and then connecting them into one or more shift registers, called **scan chains**, to provide them with external access. With external access, one can now control and observe the internal states of the digital circuit by simply shifting test stimuli into and test responses out of the shift registers during scan testing. The DFT technique has since proved to be quite effective in improving the product quality, testability, and diagnosability of scan designs [Crouch 1999; Bushnell 2000; Jha 2003; Gizopoulos 2006; Wang 2006a, 2007a]. Although scan has offered many benefits during manufacturing test, it is becoming inefficient to test deep submicron or nanometer VLSI designs. The reasons are mostly because (1) traditional test schemes that use ATPG software to target single faults have become quite expensive and (2) sufficiently high fault coverage for these deep submicron or nanometer VLSI designs is hard to sustain from the chip level to the board and system levels.

To alleviate these test problems, the scan approach is typically combined with **logic BIST** that incorporates BIST features into the scan design at the design stage [Bushnell 2000; Mourad 2000; Stroud 2002; Jha 2003]. With logic BIST, circuits that generate test patterns and analyze the output responses of the functional circuitry are embedded in the chip or elsewhere on the same board where the chip resides to test the digital logic circuit itself. Typically, pseudo-random patterns are applied to the ***circuit under test*** (CUT), while their test responses are compacted in a ***multiple-input signature register*** (MISR) [Bardell 1987; Rajski 1998; Nadeau-Dostie 2000; Stroud 2002; Jha 2003; Wang 2006a]. Logic BIST is crucial in many applications, in particular, for safety-critical and mission-critical applications. These applications, commonly found in the aerospace/defense, automotive, banking, computer, healthcare, networking, and telecommunications industries, require on-chip, on-board, or in-system self-test to improve the reliability of the entire system, as well as the ability to perform in-field diagnosis.

Since the early 2000s, **test compression**, a supplemental DFT technique to scan, is gaining industry acceptance to further reduce test data volume and test application time [Touba 2006; Wang 2006a]. Test compression involves compressing the amount of test data (both test stimulus and test response) that must be stored on ***automatic test equipment*** (ATE) for testing with a deterministic ATPG-generated test set. This is done by use of **code-based schemes** or adding additional on-chip hardware before the scan chains to decompress the test stimulus coming from the ATE and after the scan chains to compress the test response going to the ATE. This differs from logic BIST in that the test stimuli that are applied to the CUT are a deterministic (ATPG-generated) test set rather than pseudo-random patterns. Typically, test compression can provide $10\times$ to $100\times$ or even more reduction in test application time and test data volume and hence can drastically save scan test cost.

3.2 TESTABILITY ANALYSIS

Testability is a relative measure of the effort or cost of testing a logic circuit. In general, it is based on the assumption that only primary inputs and primary outputs can be directly controlled and observed, respectively. Testability reflects the effort required to perform the main test operations of controlling internal signals from primary inputs and observing internal signals at primary outputs. **Testability analysis** refers to the process of assessing the testability of a logic circuit by calculating a set of numeric measures for each signal in the circuit.

One important application of testability analysis is to assist in the decision-making process during test generation. For example, if during test generation, it is determined that the output of a certain AND gate must be set to 0, testability analysis can help decide which AND gate input is the easiest to set to 0. The conventional application is to identify areas of poor testability to guide testability enhancement, such as test point insertion, for improving the testability of the design. For this purpose, testability analysis is performed at various design stages so that testability problems can be identified and fixed as early as possible.

Since the 1970s, many testability analysis techniques have been proposed [Rutman 1972; Stephenson 1976; Breuer 1978; Grason 1979]. The ***Sandia***

Controllability/Observability Analysis Program (SCOAP) [Goldstein 1979, 1980] was the first topology-based program that populated testability analysis applications. Enhancements based on SCOAP have also been developed and used to aid in test point selection [Wang 1984, 1985]. These methods perform testability analysis by calculating the ***controllability*** and ***observability*** of each signal line, where *controllability* reflects the difficulty of setting a signal line to a required logic value from primary inputs, and *observability* reflects the difficulty of propagating the logic value of the signal line to primary outputs.

Traditionally, gate-level topologic information of a circuit is used for testability analysis. Depending on a target application, deterministic and/or random testability measures are calculated. In general, **topology-based testability analysis**, such as SCOAP or probability-based testability analysis, is computationally efficient but can produce inaccurate results for circuits containing many reconvergent fanouts. **Simulation-based testability analysis**, on the other hand, can generate more accurate estimates by simulating the circuit behavior with deterministic, random, or pseudo-random test patterns, but may require a long simulation time.

In this section, we first describe the method for performing SCOAP testability analysis. Then, probability-based testability analysis and simulation-based testability analysis are discussed.

3.2.1 **SCOAP testability analysis**

The SCOAP testability analysis program [Goldstein 1979, 1980] calculates six numeric values for each signal s in a logic circuit:

- CC0(s): Combinational 0-controllability of s
- CC1(s): Combinational 1-controllability of s
- CO(s): Combinational observability of s
- SC0(s): Sequential 0-controllability of s
- SC1(s): Sequential 1-controllability of s
- SO(s): Sequential observability of s

Roughly speaking, the three combinational testability measures, CC0, CC1, and CO, are related to the number of signals that need to be manipulated to control or observe s from primary inputs or at primary outputs, whereas the three sequential testability measures, SC0, SC1, and SO, are related to the number of clock cycles required to control or observe s from primary inputs or at primary outputs [Bushnell 2000]. The values of controllability measures range between 1 and infinite, whereas the values of observability measures range between 0 and infinite. As a boundary condition, the CC0 and CC1 values of a primary input are set to 1, the SC0 and SC1 values of a primary input are set to 0, and the CO and SO values of a primary output are set to 0.

3.2.1.1 *Combinational controllability and observability calculation*

The first step in SCOAP is to calculate the combinational controllability measures of all signals. This calculation is performed from primary inputs toward primary outputs in a breadth-first manner. More specifically, the circuit is leveled from primary inputs to primary outputs to assign a *level order* for each gate. The output controllability for each gate is then scheduled in *level order* after the controllability measures of all of its inputs have been calculated. The rules for combinational controllability calculation are summarized in Table 3.1, where a 1 is added to each rule to indicate that a signal passes through one more level of logic gate. From this table, we can see that CC0 $(s) \geq 1$ and CC1$(s) \geq 1$ for any signal *s*. A larger CC0(*s*) or CC1(*s*) value implies that it is more difficult to control *s* to 0 or 1 from primary inputs.

Once the combinational controllability measures of all signals are calculated, the combinational observability of each signal can be calculated. This calculation is also performed in a breadth-first manner while moving from primary outputs toward primary inputs. The rules for combinational observability calculation are summarized in Table 3.2, where a 1 is added to each rule to indicate that a signal passes through one more level of logic. From this table, we can see that CO$(s) \geq 0$ for any signal *s*. A larger CO(*s*) value implies that it is more difficult to observe *s* at any primary output.

Table 3.1 SCOAP Combinational Controllability Calculation Rules

	0-Controllability (Primary Input, Output, Branch)	**1-Controllability** (Primary Input, Output, Branch)
Primary Input	1	1
AND	*min* {input 0-controllabilities} + 1	Σ (input 1-controllabilities) + 1
OR	Σ (input 0-controllabilities) + 1	*min* {input 1-controllability} + 1
NOT	Input 1-controllability + 1	Input 0-controllability + 1
NAND	Σ (input 1-controllabilities) + 1	*min* {input 0-controllability} + 1
NOR	*min* {input 1-controllability) + 1	Σ (input 0-controllabilities) + 1
BUFFER	Input 0-controllability + 1	Input 1-controllability + 1
XOR	*min* {CC1(*a*) + CC1(*b*), CC0(*a*) + CC0(*b*)} + 1	*min* {CC1(*a*) + CC0(*b*), CC0(*a*) + CC1(*b*)} + 1
XNOR	*min* {CC1(*a*) + CC0(*b*), CC0(*a*) + CC1(*b*)} + 1	*min* {CC1(*a*) + CC1(*b*), CC0(*a*) + CC0(*b*)} + 1
Branch	Stem 0-controllability	Stem 1-controllability

a, *b*: inputs of an XOR or XNOR gate

Table 3.2 SCOAP Combinational Observability Calculation Rules

Observability (Primary Output, Input, Stem)	
Primary Output	0
AND/NAND	Σ (output observability, 1-controllabilities of other inputs) + 1
OR/NOR	Σ (output observability, 0-controllabilities of other inputs) + 1
NOT/BUFFER	Output observability + 1
XOR/XNOR	*a*: Σ (output observability, *min* {CC0(*b*), CC1(*b*)}) + 1 *b*: Σ (output observability, *min* {CC0(*a*), CC1(*a*)}) + 1
Stem	*min* {branch observabilities}

a, b: inputs of an XOR or XNOR gate

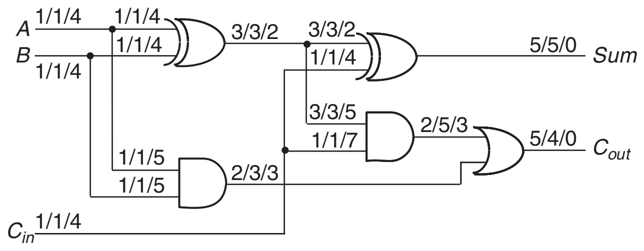


FIGURE 3.1

SCOAP full-adder example.

Figure 3.1 shows the combinational controllability and observability measures of a full-adder. The three-value tuple $v_1/v_2/v_3$ on each signal line represents the signal's 0-controllability (v_1), 1-controllability (v_2), and observability (v_3). The boundary condition is set by initializing the C0 and C1 values of the primary inputs *A, B,* and C_{in} to 1, and the CO values of the primary outputs *Sum* and C_{out} to 0. By applying the rules given in Tables 3.1 and 3.2 and starting with the given boundary condition, one can first calculate all combinational controllability measures forward and then calculate all combinational observability measures backward in *level order*.

3.2.1.2 *Sequential controllability and observability calculation*

Sequential controllability and observability measures are calculated in a similar manner as combinational measures, except that a 1 is not added as we move from one level of logic to another, but rather a 1 is added when a signal passes through a storage element. The difference is illustrated in the sequential circuit example shown in Figure 3.2, which consists of an AND gate and a positive

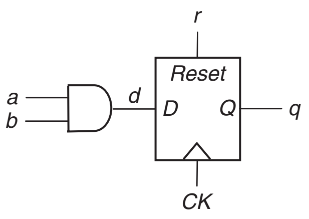


FIGURE 3.2

SCOAP sequential circuit example.

edge-triggered D flip-flop. The D flip-flop includes an active-high asynchronous reset pin *r*. SCOAP measures of a D flip-flop with a synchronous, as opposed to asynchronous, reset are shown in [Bushnell 2000].

First, we calculate the combinational and sequential controllability measures of all signals. To control signal *d* to 0, either input *a* or *b* must be set to 0. To control *d* to 1, both inputs *a* and *b* must be set to 1. Hence, the combinational and sequential controllability measures of signal *d* are:

$$\mathrm{CC0}(d) = \min\{\mathrm{CC0}(a), \mathrm{CC0}(b)\} + 1$$
$$\mathrm{SC0}(d) = \min\{\mathrm{SC0}(a), \mathrm{SC0}(b)\}$$
$$\mathrm{CC1}(d) = \mathrm{CC1}(a) + \mathrm{CC1}(b) + 1$$
$$\mathrm{SC1}(d) = \mathrm{SC1}(a) + \mathrm{SC1}(b)$$

To control the data output *q* of the D flip-flop to 0, the data input *d* and the reset signal *r* can be set to 0, while applying a rising clock edge (a 0-to-1 transition) to the clock *CK*. Alternately, this can be accomplished by setting *r* to 1 while holding *CK* at 0, without applying a clock pulse. Because a clock pulse is not applied to *CK*, a 1 is not added to the sequential controllability calculation in the second case. Therefore, the combinational and sequential 0-controllability measures of *q* are:

$$\mathrm{CC0}(q) = \min\{\mathrm{CC0}(d) + \mathrm{CC0}(CK) + \mathrm{CC1}(CK) + \mathrm{CC0}(r), \mathrm{CC1}(r) + \mathrm{CC0}(CK)\}$$
$$\mathrm{SC0}(q) = \min\{\mathrm{SC0}(d) + \mathrm{SC0}(CK) + \mathrm{SC1}(CK) + \mathrm{SC0}(r) + 1, \mathrm{SC1}(r) + \mathrm{SC0}(CK)\}$$

Here, CC0(*q*) measures how many signals in the circuit must be set to control *q* to 0, whereas SC0(*q*) measures how many flip-flops in the circuit must be clocked to set *q* to 0. To control the data output *q* of the D flip-flop to 1, the only way is to set the data input *d* to 1 and the reset signal *r* to 0, while applying a rising clock edge to the clock *CK*. Hence,

$$\mathrm{CC1}(q) = \mathrm{CC1}(d) + \mathrm{CC0}(CK) + \mathrm{CC1}(CK) + \mathrm{CC0}(r)$$
$$\mathrm{SC1}(q) = \mathrm{SC1}(d) + \mathrm{SC0}(CK) + \mathrm{SC1}(CK) + \mathrm{SC0}(r) + 1$$

Next, we calculate the combinational and sequential observability measures of all signals. The data input *d* can be observed at *q* by holding the reset signal *r* at 0 and applying a rising clock edge to *CK*. Hence,

$$\mathrm{CO}(d) = \mathrm{CO}(q) + \mathrm{CC0}(CK) + \mathrm{CC1}(CK) + \mathrm{CC0}(r)$$
$$\mathrm{SO}(d) = \mathrm{SO}(q) + \mathrm{SC0}(CK) + \mathrm{SC1}(CK) + \mathrm{SC0}(r) + 1$$

The asynchronous reset signal r can be observed by first setting q to 1, and then holding CK at the inactive state 0. Again, a 1 is not added to the sequential controllability calculation because a clock pulse is not applied to CK:

$$\mathrm{CO}(r) = \mathrm{CO}(q) + \mathrm{CC1}(q) + \mathrm{CC0}(CK)$$
$$\mathrm{SO}(r) = \mathrm{SO}(q) + \mathrm{SC1}(q) + \mathrm{SC0}(CK)$$

There are two ways to indirectly observe the clock signal CK at q: (1) set q to 1, r to 0, d to 0, and apply a rising clock edge at CK, or (2) set both q and r to 0, d to 1, and apply a rising clock edge at CK. Hence,

$$\begin{aligned}\mathrm{CO}(CK) = {}& \mathrm{CO}(q) + \mathrm{CC0}(CK) + \mathrm{CC1}(CK) + \mathrm{CC0}(r) + \\ & min\{\mathrm{CC0}(d) + \mathrm{CC1}(q), \mathrm{CC1}(d) + \mathrm{CC0}(q)\}\\ \mathrm{SO}(CK) = {}& \mathrm{SO}(q) + \mathrm{SC0}(CK) + \mathrm{SC1}(CK) + \mathrm{SC0}(r) + \\ & min\{\mathrm{SC0}(d) + \mathrm{SC1}(q), \mathrm{SC1}(d) + \mathrm{SC0}(q)\} + 1\end{aligned}$$

To observe an input of the AND gate at d requires setting the other input to 1. Therefore, the combinational and sequential observability measures for both inputs a and b are:

$$\begin{aligned}\mathrm{CO}(a) &= \mathrm{CO}(d) + \mathrm{CC1}(b) + 1\\ \mathrm{SO}(a) &= \mathrm{SO}(d) + \mathrm{SC1}(b)\\ \mathrm{CO}(b) &= \mathrm{CO}(d) + \mathrm{CC1}(a) + 1\\ \mathrm{SO}(b) &= \mathrm{SO}(d) + \mathrm{SC1}(a)\end{aligned}$$

It is important to note that controllability and observability measures calculated with SCOAP are heuristics, and only approximate the actual testability of a logic circuit. When scan design is used, testability analysis can assume that all scan cells are directly controllable and observable. It was also shown in [Agrawal 1982] that SCOAP may overestimate testability measures for circuits containing many reconvergent fanouts. However, with the capability of performing testability analysis in an $O(n)$ computational complexity for n signals in a circuit, SCOAP provides a quick estimate of the circuit's testability that can be used to guide testability enhancement and test generation.

3.2.2 Probability-based testability analysis

Topology-based testability analysis techniques, such as SCOAP, have been found to be extremely helpful in supporting test generation, which is a main topic of Chapter 14. These testability measures are able to analyze the **deterministic testability** of the logic circuit in advance and during the ATPG search process [Ivanov 1988]. On the other hand, in logic ***built-in self-test*** (BIST), which is the main topic of Section 3.4, random or pseudo-random test patterns are generated without specifically performing deterministic test pattern generation operations on any signal line. In this case, topology-based testability measures that use signal probability to analyze the **random testability** of the circuit can be used [Parker 1975; Savir 1984; Jain 1985; Seth 1985]. These measures are often referred to as **probability-based testability measures** or probability-based testability analysis techniques.

For example, given a random input pattern, one can calculate three measures for each signal *s* in a combinational circuit as follows:

- C0(*s*): Probability-based 0-controllability of *s*
- C1(*s*): Probability-based 1-controllability of *s*
- O(*s*): Probability-based observability of *s*

Here, C0(*s*) and C1(*s*) are the probability of controlling signal *s* to 0 and 1 from primary inputs, respectively. O(*s*) is the probability of observing signal *s* at primary outputs. These three probabilities range between 0 and 1. As a boundary condition, the C0 and C1 probabilities of a primary input are typically set to 0.5, and the O probability of a primary output is set to 1. For each signal *s* in the circuit, $C0(s) + C1(s) = 1$.

Many methods have been developed to calculate the probability-based testability measures. A simple method is given in the following, whose basic procedure is similar to the one used for calculating combinational testability measures in SCOAP, except that different calculation rules are used. The rules for probability-based controllability and observability calculation are summarized in Tables 3.3 and 3.4, respectively. In Table 3.3, p_0 is the initial 0-controllability chosen for a primary input, where $0 < p_0 < 1$.

Compared with SCOAP testability measures, where non-negative integers are used, probability-based testability measures range between 0 and 1. The smaller

Table 3.3 Probability-Based Controllability Calculation Rules

	0-Controllability (Primary Input, Output, Branch)	**1-Controllability** (Primary Input, Output, Branch)
Primary Input	p_0	$p_1 = 1 - p_0$
AND	1 – (output 1-controllability)	Π(input 1-controllabilities)
OR	Π(input 0-controllabilities)	1 – (output 0-controllability)
NOT	Input 1-controllability	Input 0-controllability
NAND	Π(input 1-controllabilities)	1 – (output 0-controllability)
NOR	1 – (output 1-controllability)	Π(input 0-controllabilities)
BUFFER	Input 0-controllability	Input 1-controllability
XOR	1 – 1-controllability	Σ (C1(*a*) × C0(*b*),C0(*a*) × C1(*b*))
XNOR	1 – 1-controllability	Σ (C0(*a*) × C0(*b*),C1(*a*) × C1(*b*))
Branch	Stem 0-controllability	Stem 1-controllability

a, b: inputs of an XOR or XNOR gate

Table 3.4 Probability-Based Observability Calculation Rules

Observability (Primary Output, Input, Stem)	
Primary output	1
AND/NAND	Π (output observability, 1-controllabilities of other inputs)
OR/NOR	Π (output observability, 0-controllabilities of other inputs)
NOT/BUFFER	Output observability
XOR/XNOR	*a*: Π (output observability, *max* {0-controllability of *b*, 1-controllability of *b*}) *b*: Π (output observability, *max* {0-controllability of *a*, 1-controllability of *a*})
Stem	*max* {branch observabilities}

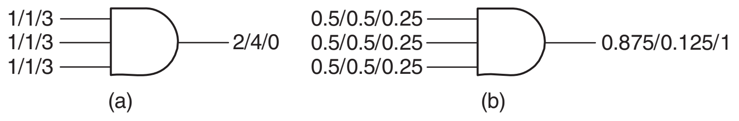


FIGURE 3.3

Comparison of SCOAP and probability-based testability measures: (a) SCOAP combinational measures. (b) Probability-based measures.

a probability-based testability measure of a signal, the more difficult it is to control or observe the signal. Figure 3.3 illustrates the difference between SCOAP testability measures and probability-based testability measures of a 3-input AND gate. The three-value tuple $v_1/v_2/v_3$ of each signal line represents the signal's 0-controllability (v_1), 1-controllability (v_2), and observability (v_3).

Signals with poor probability-based testability measures tend to be difficult to test with random or pseudo-random test patterns. The faults on these signal lines are often referred to as ***random pattern resistant*** (RP-resistant) [Savir 1984]. That is, either the probability of these signals randomly receiving a 0 or 1 from primary inputs, or the probability of observing these signals at primary outputs is low, assuming that all primary inputs have the equal probability of being set to 0 or 1.

The existence of such *RP-resistant faults* is the main reason why fault coverage that uses random or pseudo-random test patterns is low compared with the use of deterministic test patterns. In applications such as logic BIST, to solve this low fault coverage problem, test points are often inserted in the circuit to enhance the circuit's random testability. A few commonly used test point insertion techniques are discussed in [Wang 2006a].

3.2.3 Simulation-based testability analysis

In the calculation of SCOAP and probability-based testability measures as described previously, only the topologic information of a logic circuit is explicitly explored. These topology-based methods are static, in the sense that they do not use input test patterns for testability analysis. Their controllability and observability measures can be calculated in linear time, thus making them very attractive for applications that need fast testability analysis, such as test generation and logic BIST. However, the efficiency of these methods is achieved at the cost of reduced accuracy, especially for circuits that contain many reconvergent fanouts [Agrawal 1982].

As an alternative or supplement to static or topology-based testability analysis, dynamic or simulation-based methods that use input test patterns for testability analysis or testability enhancement can be performed through **statistical sampling**. Logic simulation and fault simulation techniques can be used [Bushnell 2000; Wang 2006a].

In statistical sampling, a sample set of input test patterns is selected, which is either generated randomly or derived from a given pattern set, and logic simulation is conducted to collect the responses of all or part of signal lines of interest. The commonly collected responses are the number of occurrences of 0's, 1's, 0-to-1 transitions, and 1-to-0 transitions, which are then used to profile statistically the testability of a logic circuit. These data are then analyzed to find locations of poor testability. If a signal line exhibits only a few transitions or no transitions for the sample input patterns, it might be an indication that the signal likely has poor controllability.

In addition to logic simulation, fault simulation has also been used to enhance the testability of a logic circuit with random or pseudo-random test patterns. For instance, a ***random resistant fault analysis*** (RRFA) method has been successfully applied to a high-performance microprocessor to improve the circuit's random testability in logic BIST [Rizzolo 2001]. This method is based on statistical data collected during fault simulation for a small number of random test patterns. Controllability and observability measures of each signal in the circuit are calculated by use of the probability models developed in the ***statistical fault analysis*** (STAFAN) algorithm [Jain 1985]. (STAFAN is the first method able to give reasonably accurate estimates of fault coverage in combinational circuits purely by use of input test patterns and without running fault simulation.) With these data, RRFA identifies signals that are difficult to control and/or observe, as well as signals that are statistically correlated. On the basis of the analysis results, RRFA then recommends test points to be added to the circuit to improve the circuit's random testability.

Because it can take a long simulation time to run through all input test patterns, these simulation-based methods are, in general, used to guide testability enhancement in test generation or logic BIST, when it is required to meet a very high fault coverage goal. This approach is crucial for life-critical and mission-critical applications, such as in the healthcare and defense/aerospace industries.

3.3 SCAN DESIGN

Scan design is currently the most widely used structured DFT approach. It is implemented by connecting selected storage elements of a design into one or more shift registers, called **scan chains**, to provide them with external access. Scan design accomplishes this task by replacing all selected storage elements with **scan cells**, each having one additional ***scan input*** (SI) port and one shared/additional ***scan output*** (SO) port. By connecting the SO port of one scan cell to the SI port of the next scan cell, one or more scan chains are created.

The scan-inserted design, called scan design, is now operated in three modes: **normal mode, shift mode**, and **capture mode**. Circuit operations with associated clock cycles conducted in these three modes are referred to as normal operation, shift operation, and capture operation, respectively.

In normal mode, all test signals are turned off, and the scan design operates in the original functional configuration. In both shift and capture modes, a **test mode** signal *TM* is often used to turn on all test-related fixes in compliance with scan design rules. A set of **scan design rules** that can be found in [Cheung 1997; Wang 2006a] are necessary to simplify the test, debug, and diagnose tasks, improve fault coverage, and guarantee the safe operation of the device under test. These circuit modes and operations are distinguished by use of additional test signals or test clocks. Fundamental scan architectures and at-speed clocking schemes are described in the following subsections.

3.3.1 Scan architectures

In this subsection, we first describe a few fundamental scan architectures. These fundamental scan architectures include (1) *muxed-D scan design*, in which storage elements are converted into muxed-D scan cells, (2) *clocked-scan design*, in which storage elements are converted into clocked-scan cells, and (3) *LSSD scan design*, in which storage elements are converted into *level-sensitive scan design* (LSSD) *shift register latches* (SRLs).

3.3.1.1 *Muxed-D scan design*

Figure 3.4 shows a sequential circuit example with three D flip-flops. The corresponding muxed-D full-scan circuit is shown in Figure 3.5. An edge-triggered **muxed-D scan cell** design is shown in Figure 3.5a. This scan cell is composed of a D flip-flop and a multiplexer. The multiplexer uses a ***scan enable*** (*SE*) input to select between the ***data input*** (*DI*) and the ***scan input*** (*SI*).

In normal/capture mode, *SE* is set to 0. The value present at the data input *DI* is captured into the internal D flip-flop when a rising clock edge is applied. In shift mode, *SE* is set to 1. The scan input *SI* is now used to shift in new data to the D flip-flop, while the content of the D flip-flop is being shifted out. Sample operation waveforms are shown in Figure 3.5b. The three D flip-flops,

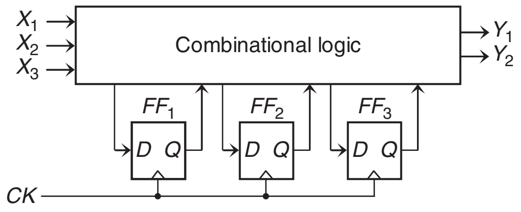


FIGURE 3.4

Sequential circuit example.

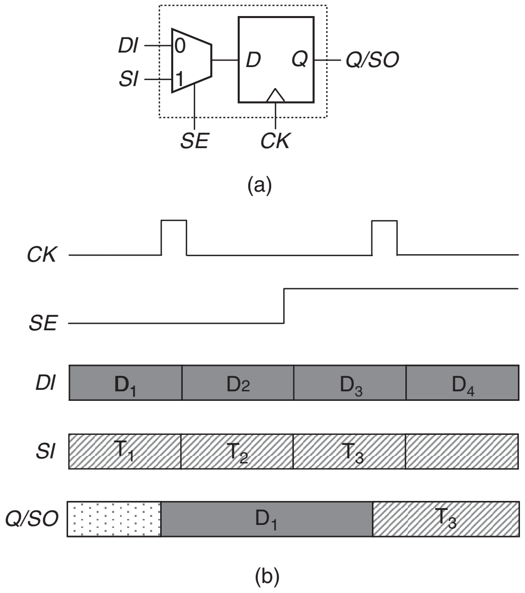


FIGURE 3.5

Edge-triggered muxed-D scan cell design and operation: (a) Muxed-D scan cell. (b) Sample waveforms.

FF_1, FF_2, and FF_3, shown in Figure 3.4, are replaced with three muxed-D scan cells, SFF_1, SFF_2, and SFF_3, respectively, shown in Figure 3.6.

In Figure 3.6, the data input *DI* of each scan cell is connected to the output of the combinational logic as in the original circuit. To form a scan chain, the scan inputs *SI* of SFF_2 and SFF_3 are connected to the outputs *Q* of the previous scan cells, SFF_1 and SFF_2, respectively. In addition, the scan input *SI* of the first scan cell SFF_1 is connected to the primary input *SI*, and the output *Q* of the last scan cell SFF_3 is connected to the primary output *SO*. Hence, in shift mode, *SE* is set to 1, and the scan cells operate as a single scan chain, which allows us to shift in any combination of logic values into the scan cells.

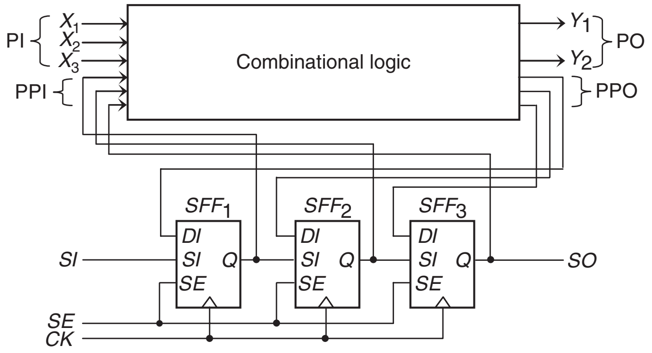


FIGURE 3.6

Muxed-D scan design.

In capture mode, *SE* is set to 0, and the scan cells are used to capture the test response from the combinational logic when a clock is applied.

In general, combinational logic in a full-scan circuit has two types of inputs: ***primary inputs*** (PIs) and ***pseudo primary inputs*** (PPIs). Primary inputs refer to the external inputs to the circuit, whereas pseudo primary inputs refer to the scan cell outputs. Both PIs and PPIs can be set to any required logic values. The only difference is that PIs are set directly in parallel from the external inputs, whereas PPIs are set serially through scan chain inputs. Similarly, the combinational logic in a full-scan circuit has two types of outputs: ***primary outputs*** (POs) and ***pseudo primary outputs*** (PPOs). Primary outputs refer to the external outputs of the circuit, whereas pseudo primary outputs refer to the scan cell inputs. Both POs and PPOs can be observed. The only difference is that POs are observed directly in parallel from the external outputs, whereas PPOs are observed serially through scan chain outputs.

3.3.1.2 ***Clocked-scan design***

An edge-triggered ***clocked-scan cell*** can also be used to replace a D flip-flop in a scan design [McCluskey 1986]. Similar to a muxed-D scan cell, a clocked-scan cell also has a data input *DI* and a scan input *SI*; however, in the clocked-scan cell, input selection is conducted with two independent clocks, data clock *DCK* and shift clock *SCK*, as shown in Figure 3.7a.

In normal/capture mode, the data clock *DCK* is used to capture the contents present at the data input *DI* into the clocked-scan cell. In shift mode, the shift clock *SCK* is used to shift in new data from the scan input *SI* into the clocked-scan cell, while the content of the clocked-scan cell is being shifted out. Sample operation waveforms are shown in Figure 3.7b.

The major advantage of the use of a clocked-scan cell is that it results in no performance degradation on the data input. A major disadvantage, however, is that it requires additional shift clock routing.

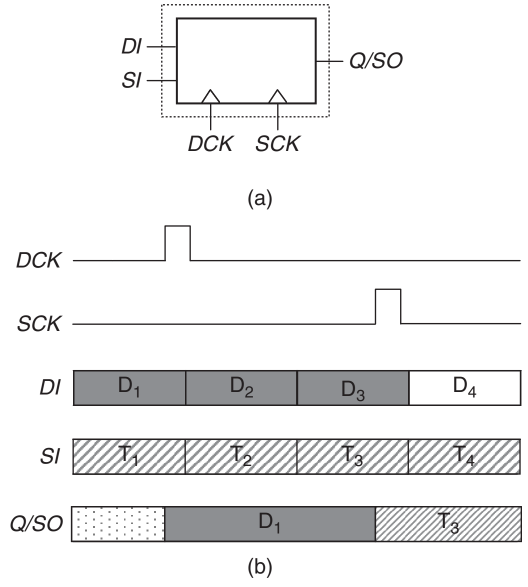


FIGURE 3.7

Clock-scan cell design and operation: (a) Clocked-scan cell. (b) Sample waveforms.

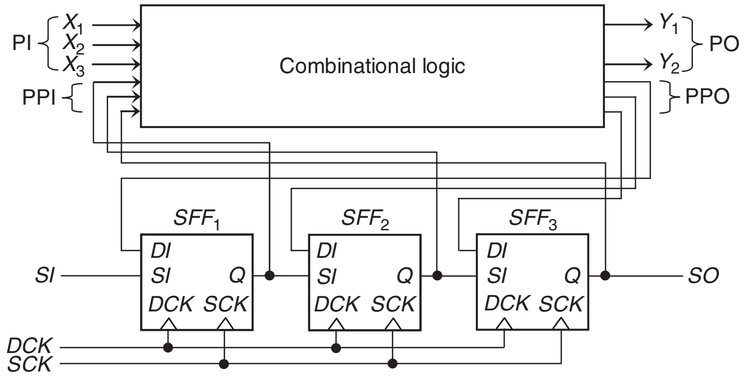


FIGURE 3.8

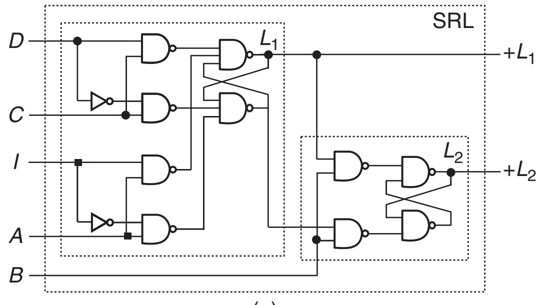
Clocked-scan design.

Figure 3.8 shows a clocked-scan design of the sequential circuit given in Figure 3.4. This clocked-scan design is tested with shift and capture operations, similar to a muxed-D scan design. The main difference is how these two operations are distinguished. In a muxed-D scan design, a scan enable signal *SE* is used, as shown in Figure 3.6. In the clocked scan shown in Figure 3.8, these two operations are distinguished by properly applying the two independent clocks *SCK* and *DCK* during shift mode and capture mode, respectively.

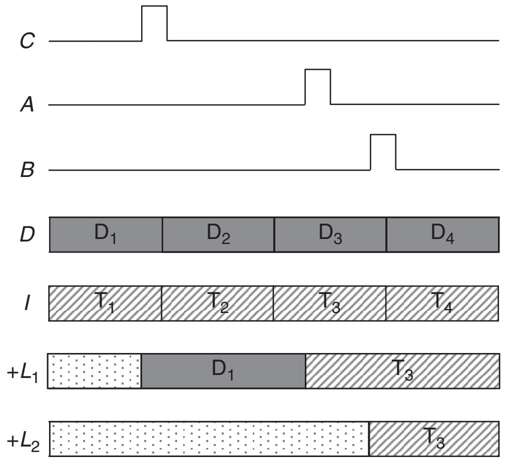
3.3.1.3 *LSSD scan design*

Figure 3.9a shows a polarity-hold ***shift register latch*** (SRL) design described in [Eichelberger 1977] that can be used as an LSSD scan cell. This scan cell contains two latches, a master two-port D latch L_1 and a slave D latch L_2. Clocks *C, A*, and *B* are used to select between the data input *D* and the scan input *I* to drive $+L_1$ and $+L_2$.

To guarantee race-free operation, clocks *A, B*, and *C* are applied in a nonoverlapping manner. In designs in which $+L_1$ is used to drive the combinational logic, the master latch L_1 uses the system clock *C* to latch system data from the data input *D* and to output this data onto $+L_1$. In designs in which $+L_2$ is



(a)



(b)

FIGURE 3.9

Polarity-hold SRL design and operation: (a) Polarity-hold SRL. (b) Sample waveforms.

used to drive the combinational logic, clock B is used after clock C to latch the system data from latch L_1 and to output these data onto $+L_2$. In both cases, capture mode uses both clocks C and B to output system data onto $+L_2$. Finally, in shift mode, clocks A and B are used to latch scan data from the scan input I and to output these data onto $+L_1$, and then latch the scan data from latch L_1 and to output these data onto $+L_2$, which is then used to drive the scan input of the next scan cell. Sample operation waveforms are shown in Figure 3.9b.

LSSD scan designs can be implemented with either a **single-latch design** or a **double-latch design**. In single-latch design [Eichelberger 1977], the output port $+L_1$ of the master latch L_1 is used to drive the combinational logic of the design. In this case, the slave latch L_2 is used only for scan testing. Because LSSD designs use latches instead of flip-flops, at least two system clocks C_1 and C_2 are required to prevent combinational feedback loops from occurring. In this case, combinational logic driven by the master latches of the first system clock C_1 are used to drive the master latches of the second system clock C_2, and vice versa. For this to work, the system clocks C_1 and C_2 should be applied in a nonoverlapping fashion. Figure 3.10a shows an LSSD single-latch design with the polarity-hold SRL shown in Figure 3.9.

Figure 3.10b shows an example of LSSD **double-latch design** [DasGupta 1982]. In normal mode, the C_1 and C_2 clocks are used in a nonoverlapping manner, where the C_2 clock is the same as the B clock. The testing of an LSSD scan design is conducted with shift and capture operations, similar to a muxed-D scan design. The main difference is how these two operations are distinguished. In a muxed-D scan design, a scan enable signal SE is used, as shown in Figure 3.6. In an LSSD scan design, these two operations are distinguished by properly applying nonoverlapping clock pulses to clocks C_1, C_2, A, and B. During the shift operation, clocks A and B are applied in a nonoverlapping manner, and the scan cells $SRL_1 \sim SRL_3$ form a single scan chain from SI to SO. During the capture operation, clocks C_1 and C_2 are applied in a nonoverlapping manner to load the test response from the combinational logic into the scan cells.

The major advantage of the use of an LSSD scan cell is that it allows us to insert scan into a latch-based design. In addition, designs that use LSSD are guaranteed to be race-free, which is not the case for muxed-D scan and clocked-scan designs. A major disadvantage, however, is that it requires routing for the additional clocks, which increases routing complexity.

The operation of a polarity-hold SRL is race-free if clocks C and B, as well as A and B, are nonoverlapping. This characteristic is used to implement LSSD circuits that are guaranteed to have race-free operation in normal mode and in test mode.

3.3.2 At-speed testing

Although scan design is commonly used in the industry for slow-speed stuck-at fault testing, its real value is in providing **at-speed testing** for high-speed and

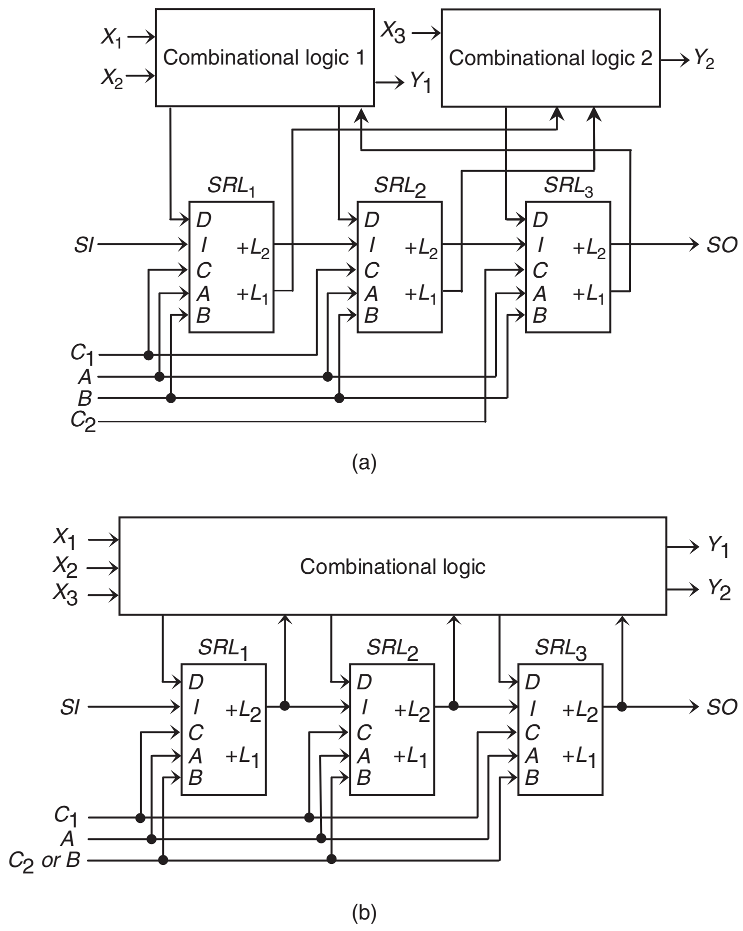


FIGURE 3.10

LSSD designs: (a) LSSD single-latch design. (b) LSSD double-latch design.

high-performance circuits. These circuits often contain multiple clock domains, each running at an operating frequency that is either synchronous or asynchronous to the other clock domains. Two clock domains are said to be **synchronous** if the active edges of both clocks controlling the two clock domains can be aligned precisely or triggered simultaneously. Two clock domains are said to be **asynchronous** if they are not synchronous.

There are two basic capture-clocking schemes for testing multiple clock domains at-speed: (1) *skewed-load* [Savir 1993] (also called *launch-on-shift* [LOS]) and (2) *double-capture* [Wang 2006a] (also called *launch-on-capture* [LOC] or *broad-side* [Savir 1994]). Both schemes can be used to test path-delay faults and transition faults within each clock domain (called **intra-clock-domain**

faults) or across clock domains (called **inter-clock-domain faults**). Skewed-load uses the last shift clock pulse followed immediately by a capture clock pulse to launch the transition and capture the output test response, respectively. Double-capture uses two consecutive capture clock pulses to launch the transition and capture the output test response, respectively. In both schemes, both launch and capture clock pulses must be running at the domain's operating speed or at-speed. The difference is that skewed-load requires the domain's scan enable signal *SE* to switch its value between the launch and capture clock pulses making *SE* act as a clock signal. Figure 3.11 shows sample waveforms that use the basic skewed-load and double-capture at-speed test schemes.

Scan designs typically include a few clock domains that will interact with one another. To guarantee the success of the capture operation, additional care must be taken in terms of the way the capture clocks are applied. This is mainly because the clock skew between different clock domains is typically large. To prevent this from happening, clocks can be applied sequentially (with the **staggered clocking** scheme [Wang 2005a, 2007b]), such that any clock skew that exists between the clock domains can be tolerated during the test generation process. It is also possible to apply only one clock during each capture operation by use of the **one-hot clocking** scheme. Most modern ATPG programs used currently can also automatically mask off unknown values (*X*'s) at the originating scan cells or receiving scan cells across clock domains. In this case, all clocks can also be applied simultaneously with the **simultaneous clocking** scheme [Wang 2007b]. During simultaneous clocking, if the launch clock pulses [Rajski 2003; Wang 2006a] or the capture clock pulses [Nadeau-Dostie 1994; Wang 2006a] can be aligned precisely, which applies only for synchronous clock domains, then the **aligned clocking** scheme can be used, and there is no need to mask off unknown values across these synchronous clock domains. These clocking schemes are illustrated in Figure 3.12.

In general, one-hot clocking produces the highest fault coverage at the expense of generating many more test patterns than other schemes. Simultaneous clocking can generate the smallest number of test patterns but may result

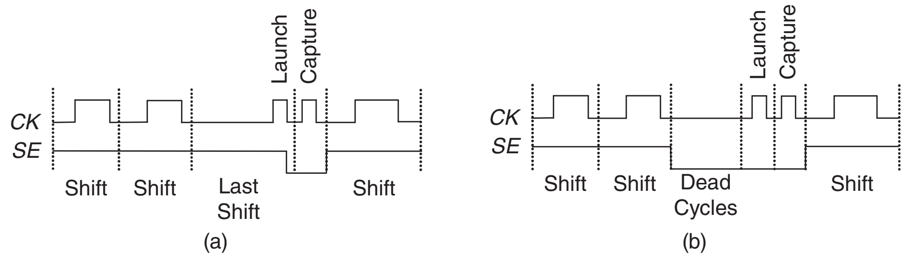


FIGURE 3.11

Basic at-speed test schemes: (a) Skewed-load. (b) Double-capture.

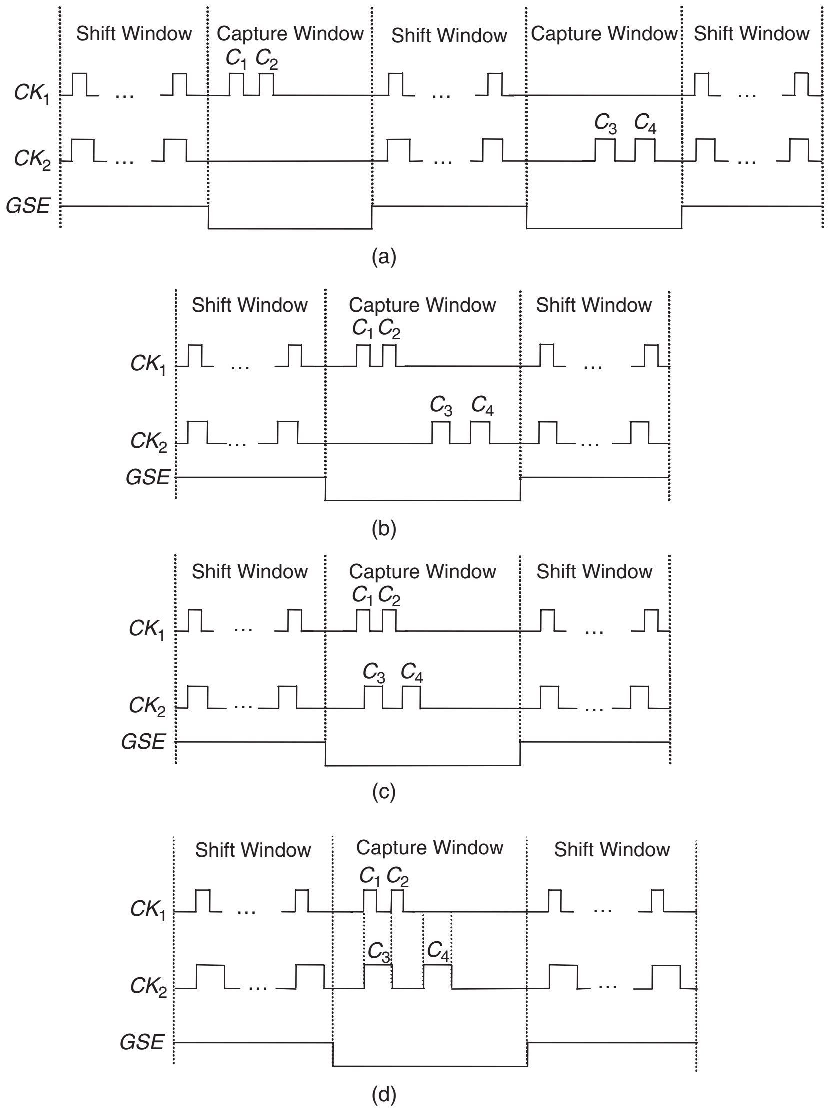


FIGURE 3.12

At-speed clocking schemes for testing two interacting clock domains: (a) One-hot clocking. (b) Staggered clocking. (c) Simultaneous clocking. (d) Aligned clocking.

in high fault coverage loss because of unknown (*X*) masking. The staggered clocking scheme is a happy medium because of its ability to generate test pattern count close to simultaneous clocking and fault coverage close to one-hot clocking. For large designs, it is no longer uncommon for transition fault ATPG to take more than 2 to 4 weeks to complete. To reduce test generation time while at the same time obtaining the highest fault coverage, modern ATPG programs tend to either (1) run simultaneous clocking followed by one-hot clocking or (2) use staggered clocking followed by one-hot clocking. As a result, modern **at-speed scan architectures** now start supporting a combination of at-speed clocking schemes for test circuits comprising multiple synchronous and asynchronous clock domains. Some programs can even generate test patterns by mixing skewed-load and double-capture schemes.

3.4 LOGIC BUILT-IN SELF-TEST

Logic built-in self-test (BIST) requires using a portion of the circuit to test itself on-chip, on-board, or in-system. A typical logic BIST system is illustrated in Figure 3.13. The ***test pattern generator*** (TPG) automatically generates test patterns for application to the inputs of the ***circuit under test*** (CUT). The ***output response analyzer*** (ORA) automatically compacts the output responses of the CUT into a *signature*. Specific BIST timing control signals, including scan enable signals and clocks, are generated by the **logic BIST controller** for coordinating the BIST operation among the TPG, CUT, and ORA. The logic BIST controller provides a pass/fail indication once the BIST operation is complete. It includes comparison logic to compare the *final signature* with an embedded *golden signature*, and often comprises **diagnostic logic** for fault diagnosis. Because compaction is commonly used for output response analysis, it is required that all storage elements in the TPG, CUT, and ORA be initialized to known states before self-test, and no unknown (X) values are allowed to propagate from the CUT to the ORA. In other words, the CUT must comply with more stringent **BIST-specific design rules** [Wang 2006a] in addition to those *scan design rules* required for scan design.

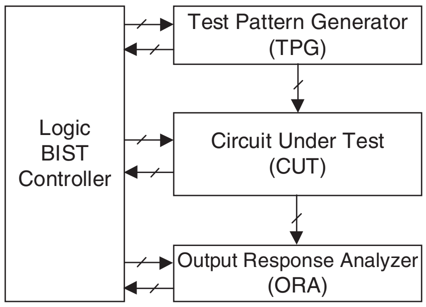


FIGURE 3.13

A typical logic BIST system.

3.4.1 **Test pattern generation**

For logic BIST applications, in-circuit TPGs constructed from ***linear feedback shift registers*** (**LFSRs**) are most commonly used to generate test patterns or test sequences for exhaustive testing, pseudo-random testing, and pseudo-exhaustive testing.

Exhaustive testing always guarantees 100% single-stuck and multiple-stuck fault coverage. This technique requires all possible 2^n test patterns to be applied to an n-input combinational CUT, which can take too long for combinational circuits where n is huge. Therefore, **pseudo-random testing** [Bardell 1987] is often used for generating a subset of the 2^n test patterns and uses fault simulation to calculate the exact fault coverage. In some cases, this might become quite time-consuming, if not infeasible. To eliminate the need for fault simulation while at the same time maintaining 100% single-stuck fault coverage, we can use **pseudo-exhaustive testing** [McCluskey 1986] to generate 2^w or $2^k - 1$ test patterns, where $w < k < n$, when each output of the n-input combinational CUT at most depends on w inputs. For testing delay faults, hazards must also be taken into consideration.

Standard LFSR

Figure 3.14 shows an n-stage **standard LFSR**. It consists of n D flip-flops and a selected number of *exclusive-OR* (XOR) gates. Because XOR gates are placed on the external feedback path, the standard LFSR is also referred to as an **external-XOR LFSR** [Golomb 1982].

Modular LFSR

Similarly, an n-stage **modular LFSR** with each XOR gate placed between two adjacent D flip-flops, as shown in Figure 3.15, is referred to as an **internal-XOR LFSR** [Golomb 1982]. The modular LFSR runs faster than its corresponding standard LFSR, because each stage introduces at most one XOR-gate delay.

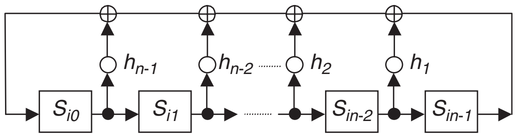


FIGURE 3.14

An n-stage (external-XOR) standard LFSR.

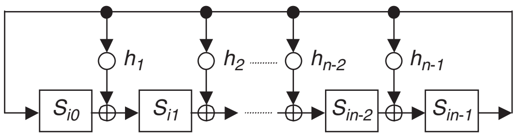


FIGURE 3.15

An n-stage (internal-XOR) modular LFSR.

LFSR Properties

The internal structure of the *n*-stage LFSR in each figure can be described by specifying a **characteristic polynomial** of degree *n*, *f*(*x*), in which the symbol h_i is either 1 or 0, depending on the existence or absence of the feedback path, where

$$f(x) = 1 + h_1x + h_2x^2 + \ldots + h_{n-1}x^{n-1} + x^n$$

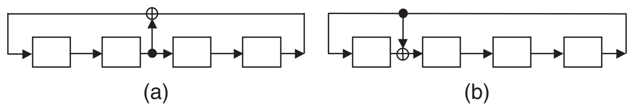
Let S_i represent the contents of the *n*-stage LFSR after *i*th shifts of the initial contents, S_0, of the LFSR, and $S_i(x)$ be the polynomial representation of S_i. Then, $S_i(x)$ is a polynomial of degree $n-1$, where

$$S_i(x) = S_{i0} + S_{i1}x + S_{i2}x^2 + \ldots + S_{in\text{-}2}x^{n-2} + S_{in-1}x^{n-1}$$

If *T* is the smallest positive integer such that *f*(*x*) divides $1 + x^T$, then the integer *T* is called the **period** of the LFSR. If $T = 2^n-1$, then the *n*-stage LFSR generating the **maximum-length sequence** is called a **maximum-length LFSR**.

For example, consider the four-stage standard and modular LFSRs shown in Figures 3.16a and 3.16b below. The characteristic polynomials, *f*(*x*), used to construct both LFSRs are $1 + x^2 + x^4$ and $1 + x + x^4$, respectively.

The test sequences generated by each LFSR, when its initial contents, S_0, are set to {0001} or $S_0(x) = x^3$, are listed in Figures 3.16c and 3.16d, respectively.



0 0 0 1	**0 0 0 1**
1 0 0 0	1 1 0 0
0 1 0 1	0 1 1 0
1 0 1 0	0 0 1 1
0 1 0 1	1 1 0 1
0 0 1 0	1 0 1 0
0 0 0 1	0 1 0 1
1 0 0 0	1 1 1 0
0 1 0 1	0 1 1 1
1 0 1 0	1 1 1 1
0 1 0 1	1 0 1 1
0 0 1 0	1 0 0 1
0 0 0 1	1 0 0 0
1 0 0 0	0 1 0 0
0 1 0 0	0 0 1 0
1 0 1 0	**0 0 0 1**
(c)	(d)

FIGURE 3.16

Example four-stage test pattern generators (TPGs): (a) Four-stage standard LFSR. (b) Four-stage modular LFSR. (c) Test sequence generated by (a). (d) Test sequence generated by (b).

Because the first test sequence repeats after 6 patterns and the second test sequence repeats after 15 patterns, the LFSRs have periods of 6 and 15, respectively. This further implies that $1 + x^6$ can be divided by $1 + x^2 + x^4$, and $1 + x^{15}$ can be divided by $1 + x + x^4$.

Define a **primitive polynomial** of degree n over **Galois field** $GF(2)$, $p(x)$, as a polynomial that divides $1 + x^T$, but not $1 + x^i$, for any integer $i < T$, where $T = 2^n - 1$ [Golomb 1982]. A primitive polynomial is **irreducible**. Because $T = 15 = 2^4 - 1$, the characteristic polynomial, $f(x) = 1 + x + x^4$, used to construct Figure 3.16b is a primitive polynomial, and thus the modular LFSR is a maximum-length LFSR. Let

$$r(x) = f(x)^{-1} = x^n f(x^{-1})$$

Then $r(x)$ is defined as a **reciprocal polynomial** of $f(x)$ [Peterson 1972]. A reciprocal polynomial of a primitive polynomial is also a primitive polynomial. Thus, the reciprocal polynomial of $f(x) = 1 + x + x^4$ is also a primitive polynomial, with $p(x) = r(x) = 1 + x^3 + x^4$.

Table 3.5 lists a set of primitive polynomials of degree n up to 100. It was taken from [Bardell 1987]. A different set was given in [Wang 1988a]. Each polynomial can be used to construct minimum-length LFSRs in standard or modular form. For primitive polynomials of degree up to 300, consult [Bardell 1987].

3.4.1.1 *Exhaustive testing*

Exhaustive testing requires applying 2^n exhaustive patterns to an n-input combinational CUT. Any **binary counter** can be used as an ***exhaustive pattern generator*** (EPG) for this purpose. Figure 3.17 shows an example of a 4-bit binary counter design for testing a 4-input combinational CUT.

Exhaustive testing guarantees that all detectable, combinational faults (those that do not change a combinational circuit into a sequential circuit) will be detected. This approach is especially useful for circuits in which the number of inputs, n, is a small number (*e.g.*, 20 or less). When n is larger than 20, the test time may be prohibitively long and is thus not recommended. The following techniques are aimed at reducing the number of test patterns. They are recommended when exhaustive testing is impractical.

3.4.1.2 *Pseudo-random testing*

One approach, which can reduce test length but sacrifices the circuit's fault coverage, uses a ***pseudo-random pattern generator*** (PRPG) for generating a pseudo-random sequence of test patterns [Bardell 1987; Rajski 1998; Bushnell 2000; Jha 2003]. **Pseudo-random testing** has the advantage of being applicable to both sequential and combinational circuits; however, there are difficulties in determining the required test length and fault coverage.

Table 3.5 Primitive Polynomials of Degree *n* up to 100

n	Exponents	*n*	Exponents	*n*	Exponents	*n*	Exponents
1	0	26	8 7 1 0	51	16 15 1 0	76	36 35 1 0
2	1 0	27	8 7 1 0	52	3 0	77	31 30 1 0
3	1 0	28	3 0	53	16 15 1 0	78	20 19 1 0
4	1 0	29	2 0	54	37 36 1 0	79	9 0
5	2 0	30	16 15 1 0	55	24 0	80	38 37 1 0
6	1 0	31	3 0	56	22 21 1 0	81	4 0
7	1 0	32	28 27 1 0	57	7 0	82	38 35 3 0
8	6 5 1 0	33	13 0	58	19 0	83	46 45 1 0
9	4 0	34	15 14 1 0	59	22 21 1 0	84	13 0
10	3 0	35	2 0	60	1 0	85	28 27 1 0
11	2 0	36	11 0	61	16 15 1 0	86	13 12 1 0
12	7 4 3 0	37	12 10 2 0	62	57 56 1 0	87	13 0
13	4 3 1 0	38	6 5 1 0	63	1 0	88	72 71 1 0
14	12 11 1 0	39	4 0	64	4 3 1 0	89	38 0
15	1 0	40	21 19 2 0	65	18 0	90	19 18 1 0
16	5 3 2 0	41	3 0	66	10 9 1 0	91	84 83 1 0
17	3 0	42	23 22 1 0	67	10 9 1 0	92	13 12 1 0
18	7 0	43	6 5 1 0	68	9 0	93	2 0
19	6 5 1 0	44	27 26 1 0	69	29 27 2 0	94	21 0
20	3 0	45	4 3 1 0	70	16 15 1 0	95	11 0
21	2 0	46	21 20 1 0	71	6 0	96	49 47 2 0
22	1 0	47	5 0	72	53 47 6 0	97	6 0
23	5 0	48	28 27 1 0	73	25 0	98	11 0
24	4 3 1 0	49	9 0	74	16 15 1 0	99	47 45 2 0
25	3 0	50	27 26 1 0	75	11 10 1 0	100	37 0

Note: "24 4 3 1 0" means $p(x) = x^{24} + x^4 + x^3 + x^1 + x^0 = x^{24} + x^4 + x^3 + x + 1$.

3.4.1.2.1 Maximum-length LFSR

Maximum-length LFSRs are commonly used for pseudo-random pattern generation. Each LFSR produces a sequence with 0.5 probability of generating 1's

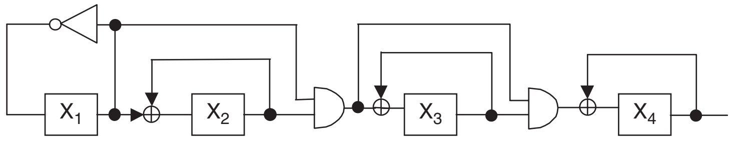


FIGURE 3.17

Example binary counter as EPG.

(or with probability distribution 0.5) at every output. The **LFSR pattern generation technique** that uses these LFSRs, in standard or modular form, to generate patterns for the entire design has the advantage of being very easy to implement. The major problem with this approach is that some circuits may be ***random pattern resistant*** (RP-resistant). For instance, consider a 5-input OR gate. The probability of applying an all-zero pattern to all inputs is 1/32. This makes it difficult to test the RP-resistant OR-gate output stuck-at-1.

3.4.1.2.2 Weighted LFSR

It is possible to increase fault coverage (and detect most RP-resistant faults) in RP-resistant designs. A **weighted pattern generation technique** that uses an LFSR and a combinational circuit was first described in [Schnurmann 1975]. The combinational circuit inserted between the output of the LFSR and the CUT is to increase the frequency of occurrence of one logic value while decreasing the other logic value. This approach may increase the probability of detecting those faults that are hard to detect with the typical LFSR pattern generation technique.

Implementation methods for realizing this scheme are further discussed in [Chin 1984]. The weighted pattern generation technique described in that paper modifies the maximum-length LFSR to produce an equally weighted distribution of 0's and 1's at the input of the CUT. It skews the LFSR probability distribution of 0.5 to either 0.25 or 0.75 to increase the chance of detecting those faults that are hard to detect with just a 0.5 distribution. Better fault coverage was also found in [Wunderlich 1987], where probability distributions in a multiple of 0.125 (rather than 0.25) are used. Figure 3.18 shows a four-stage weighted (maximum-length) LFSR with probability distribution 0.25 [Chin 1984].

3.4.1.2.3 Cellular automata

Cellular automata were first introduced in [Wolfram 1983]. They yielded better randomness property than LFSRs [Hortensius 1989]. The *cellular automaton based* (or CA-based) *pseudo-random pattern generator* (PRPG) is attractive for BIST applications [Khara 1987; Gloster 1988; Wang 1989; van Sas 1990] because it (1) provides patterns that look *more* random at the circuit inputs, (2) has higher opportunity to reach very high fault coverage in a circuit that is RP-resistant, and

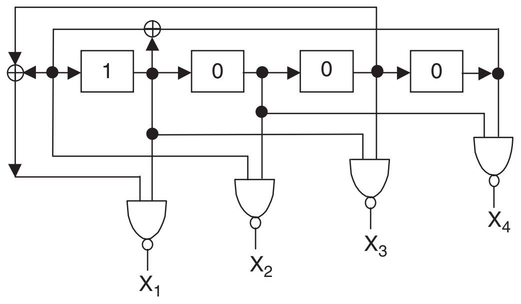


FIGURE 3.18

Example weighted LFSR as PRPG.

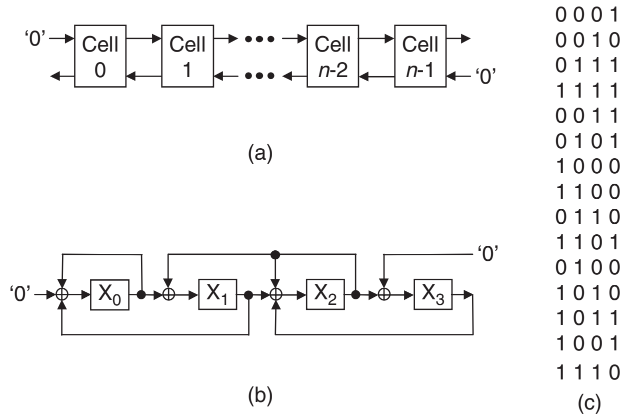


FIGURE 3.19

Example cellular automaton (CA) as PRPG: (a) General structure of an *n*-stage CA. (b) Four-stage CA. (c) Test sequence generated by (b).

(3) has implementation advantages because it only requires adjacent neighbor communication (no global feedback unlike the modular LFSR case).

A *cellular automaton* (CA) is a collection of **cells** with forward and backward connections. A general structure is shown in Figure 3.19a. Each cell can only connect to its local neighbors (adjacent left and right cells). The connections are expressed as **rules**; each rule determines the next state of a cell on the basis of the state of the cell and its neighbors. Assume cell *i* can only talk with its neighbors, $i - 1$ and $i + 1$. Define:

$$\textit{Rule 90}: x_i(t+1) = x_{i-1}(t) + x_{i+1}(t)$$

and

$$\textit{Rule 150}: x_i(t+1) = x_{i-1}(t) + x_i(t) + x_{i+1}(t)$$

Then the two rules, *rule 90* and *rule 150*, can be established on the basis of the following state transition table:

$x_{i-1}(t)x_i(t)x_{i+1}(t)$	111	110	101	100	011	010	001	000
Rule 90: $x_i(t+1)$	0	1	0	1	1	0	1	0
	2^6 +	2^4 +	2^3 +	$2^1 = 90$				
Rule 150: $x_i(t+1)$	1	0	0	1	0	1	1	0
	2^7 +	2^4 +	2^2 +	$2^1 = 150$				

The terms *rule 90* and *rule 150* were derived from their decimal equivalents of the binary code for the next state of cell *i* [Hortensius 1989]. Figure 3.19b shows an example of a four-stage CA generated by alternating rules 150 (on even cells) and 90 (on odd cells). Similar to the four-stage modular LFSR given in Figure 3.16b, the four-stage CA generates a *maximum-length sequence* of 15 distinct states as listed in Figure 3.19c.

It has been shown in [Hortensius 1989] that by combining cellular automata rules 90 and 150, an *n*-stage CA can generate a maximum-length sequence of $2^n - 1$. The construction rules for $4 \le n \le 53$ can be found in [Hortensius 1989] and are listed in Table 3.6.

The CA-based PRPG can be programmed as a **universal CA** for generating different orders of test sequences. A **universal CA-cell** for generating patterns on the basis of *rule 90* or *rule 150* is given in Figure 3.20 [Wang 1989]. When the RULE150_SELECT signal is set to 1, the universal CA-cell will behave as a *rule 150* cell; otherwise, it will act as a *rule 90* cell. This *universal CA* structure is useful for BIST applications where it is required to obtain very high fault coverage for RP-resistant designs or detect additional classes of faults.

3.4.1.3 ***Pseudo-exhaustive testing***

Another approach to reduce the test time to a practical value while retaining many of the advantages of exhaustive testing is the **pseudo-exhaustive test technique**. It applies fewer than 2^n test patterns to an *n*-input combinational CUT. The technique depends on whether any output is driven by all of its inputs. If none of the outputs depends on all inputs, a **verification test approach** proposed in [McCluskey 1984] can be used to test these circuits. In circuits in which there is one output that depends on all inputs or the test time that uses verification testing is still too long, a **segmentation test approach** must be used [McCluskey 1981]. Pseudo-exhaustive testing guarantees single-stuck fault coverage without any detailed circuit analysis.

Table 3.6 Construction Rules for Cellular Automat of Length *n* up to 53

n	Rule*	*n*	Rule*
4	05	29	2,512,712103
5	31	30	7,211,545,075
6	25	31	04,625,575,630
7	152	32	10,602,335,725
8	325	33	03,047,162,605
9	625	34	036,055,030,672
10	0,525	35	127,573,165,123
11	3,252	36	514,443,726,043
12	2,252	37	0,226,365,530,263
13	14,524	38	0,345,366,317,023
14	17,576	39	6,427,667,463,554
15	44,241	40	00,731,257,441,345
16	152,525	41	15,376,413,143,607
17	175,763	42	11,766,345,114,746
18	252,525	43	035,342,704,132,622
19	0,646,611	44	074,756,556,045,302
20	3,635,577	45	151,315,510,461,515
21	3,630,173	46	0,112,312,150,547,326
22	05,252,525	47	0,713,747,124,427,015
23	32,716,532	48	0,606,762,247,217,017
24	77,226,526	49	02,675,443,137,056,631
25	136,524,744	50	23,233,006,150,544,226
26	132,642,730	51	04,135,241,323,505,027
27	037,014,415	52	031,067,567,742,172,706
28	0,525,252,525	53	207,121,011,145,676,625

**Rule is given in octal format. For n* = 7, Rule = 152 = 001,101,010 = 1,101,010, where "0" denotes a *rule 90* cell and "1" denotes a *rule 150* cell, or vice versa.

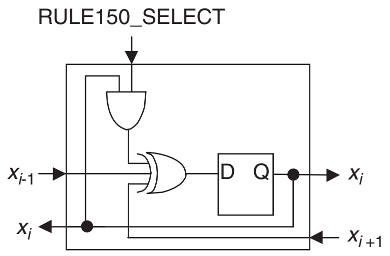


FIGURE 3.20

A universal CA-cell structure.

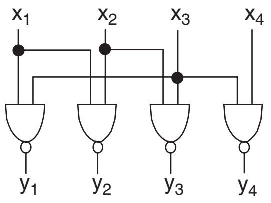


FIGURE 3.21

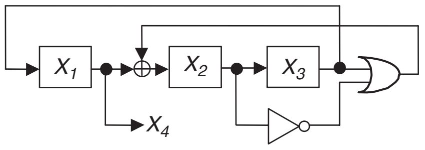
An $(n,w) = (4,2)$ CUT.

Verification testing [McCluskey 1984] divides the circuit under test into m cones, where m is the number of outputs. It is based on backtracing from each circuit output to determine the actual number of inputs that drive the output. Each cone will receive exhaustive test patterns, and all cones are tested concurrently.

Assume the combinational CUT has n inputs and m outputs. Let w be the maximum number of input variables on which any output of the CUT depends. Then, the n-input m-output combinational CUT is defined as an (n,w) CUT, where $w < n$. Figure 3.21 shows an $(n,w) = (4,2)$ CUT that will be used as an example for designing the *pseudo-exhaustive pattern generators* (PEPGs).

3.4.1.3.1 Syndrome driver counter

The first method for **pseudo-exhaustive pattern generation** was proposed in [Savir 1980]. ***Syndrome driver counters*** (SDCs) are used to generate test patterns [Barzilai 1981]. The SDC can be a binary counter, a maximum-length LFSR, or a complete LFSR. This method checks whether some circuit inputs can share the same test signal. If n-p inputs, $p < n$, can share the **test signals** with the other p inputs, then the circuit can be tested exhaustively with these p inputs. In this case, the test length becomes 2^p if $p = w$, or $2^p - 1$ if $p > w$. Figure 3.22 shows a three-stage SDC used to test the circuit given in Figure 3.21. Because both inputs x_1 and x_4 do



```
0 0 1 0
1 0 0 1
1 1 0 1
1 1 1 1
0 1 1 0
1 0 1 1
0 1 0 0
```

FIGURE 3.22

Example syndrome driver counter as PEPG.

not drive the same output, one test signal can be used to drive both inputs. In this case, p is 3, and the test length becomes $2^3 - 1 = 7$. Designs based on the SDC method for in-circuit test pattern generation are simple. The problem with this method is that when p is close to n, it may still take too long to test the circuit.

3.4.1.3.2 Condensed LFSR

The problem can be solved by use of the **condensed LFSR** approach proposed in [Wang 1986a]. Condensed LFSRs are constructed on the basis of **linear codes** [Peterson 1972]. An (n,k) *linear code over GF*(2) generates a code space C containing 2^k distinct code words (n-tuples) with the following property: if $c_1 \in C$ and $c_2 \in C$, then $c_1 + c_2 \in C$. Define an (n,k) *condensed LFSR* as an n-stage modular LFSR with period 2^k-1. A condensed LFSR for testing an (n,w) CUT is constructed by first computing the smallest integer k such that:

$$w \le \lceil k/(n-k+1) \rceil + \lfloor k/(n-k+1) \rfloor$$

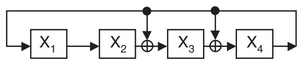
where $\lceil x \rceil$ denotes the smallest integer equal to or greater than the real number x, and $\lfloor y \rfloor$ denotes the largest integer equal to or smaller than the real number y.

Then, by use of:

$$f(x) = g(x)p(x) = (1 + x + x^2 + \ldots + x^{n-k})p(x)$$

an (n,k) *condensed LFSR* can be realized, where $g(x)$ is a **generator polynomial** of degree n-k generating the (n,k) linear code, and $p(x)$ is a primitive polynomial of degree k.

Consider the $(n,k) = (4,3)$ condensed LFSR shown in Figure 3.23a used to test the $(n,w) = (4,2)$ CUT. Because $n = 4$ and $w = 2$, we obtain $k = 3$ and



(a)

```
1 1 0 0
0 1 1 0
0 0 1 1
1 0 1 0
0 1 0 1
1 0 0 1
1 1 1 1
```

(b)

FIGURE 3.23

Example condensed LFSR as PEPG: (a) (4,3) condensed LFSR. (b) Test sequence generated by (a).

$(n - k) = 1$. Selecting $p(x) = 1 + x + x^3$, we have $f(x) = (1 + x)(1 + x + x^3) = 1 + x^2 + x^3 + x^4$. Figure 3.23b lists the generated period-7 test sequence. It is important to note that the seed polynomial $S_0(x)$ of the LFSR must be divisible by $g(x)$. In the example, we set $S_0(x) = g(x) = 1 + x$, or S_0 to {1100}.

For any given (n,w) CUT, this method uses at most two seeds and has shown to be effective when $w \geq n/2$. Designs based on this method are simple. However, this technique uses more patterns than the **combined LFSR/SR** approach, which uses a combination of an LFSR and a *shift register* (SR) [Barzilai 1983; Tang 1984; Chen 1987] and the **cyclic LFSR** approach [Wang 1987, 1988b] when $w < n/2$. For other verification test approaches, refer to [Abramovici 1994; Wang 2006a].

3.4.2 **Output response analysis**

For scan designs, our assumption was that output responses coming out of the *circuit under test* (CUT) are compared directly on a tester. For BIST operations, it is impossible to store all output responses on-chip, on-board, or in-system to perform bit-by-bit comparison. An *output response analysis* technique must be used such that output responses can be compacted into a **signature** and compared with a *golden signature* for the fault-free circuit either embedded on-chip or stored off-chip.

Compaction differs from *compression* in that compression is loss-less, whereas compaction is lossy. **Compaction** is a method for dramatically reducing the number of bits in the original circuit response during testing in which some information is lost. **Compression** is a method for reducing the number of bits in the original circuit response in which no information is lost, such that the original output sequence can be fully regenerated from the compressed sequence [Bushnell 2000]. Because all output response analysis schemes involve information loss, they are referred to as *output response compaction*. However, there is no general consensus in academia yet as to when the terms compaction or compression are to be used. However, for output response analysis, throughout the book, we will refer to the lossy compression as compaction.

In this section, we will present three different output response compaction techniques: (1) **ones count testing**, (2) **transition count testing**, and (3) **signature analysis**. We will also describe the architectures of the *output response analyzers* (ORAs) that are used. The signature analysis technique will be described in more detail, because it is the most popular compaction technique in use today.

When compaction is used, it is important to ensure that the faulty and fault-free signatures are different. If they are the same, the fault(s) can go undetected. This situation is referred to as **error masking**, and the erroneous output response is said to be an **alias** of the correct output response [Abramovici 1994]. It is also important to ensure that none of the output responses contains an unknown (X) value. If an unknown value is generated and propagated directly or indirectly to the ORA, then the ORA can no longer function reliably. Therefore, it is required that all unknown (X) propagation problems be fixed to ensure that the logic BIST system will operate correctly. Such **X-blocking** or **X-bounding** techniques have been extensively discussed in [Wang 2006a].

3.4.2.1 *Ones count testing*

Assume that the CUT has only one output and the output contains a stream of L bits. Let the fault-free output response, R_0, be $\{r_0\, r_1\, r_2 \ldots r_{L-1}\}$. The **ones count test technique** will only need a counter to count the number of 1's in the bit stream. For instance, if $R_0 = \{0101100\}$, then the signature or ones count of R_0, $OC(R_0)$, is 3. If fault f_1 present in the CUT causes an erroneous response $R_1 = \{1100110\}$, then it will be detected because $OC(R_1) = 4$. However, fault f_2 causing $R_2 = \{0101010\}$ will not be detected because $OC(R_2) = OC(R_0) = 3$. Let the fault-free signature or ones count be m. There will be $C(L,m)$ possible ways having m 1's in an L-bit stream. Assuming all faulty sequences are equally likely to occur as the response of the CUT, the **aliasing probability** or **masking probability** of the use of ones count testing having m 1's [Savir 1985] can be expressed as

$$P_{OC}(m) = \left(C(L,m) - 1\right)/(2^L - 1)$$

In the previous example, where $m = OC(R_0) = 3$ and $L = 7$, $P_{OC}(m) = 34/127 = 0.27$. Figure 3.24 shows the ones count test circuit for testing the CUT with T patterns. The number of stages in the counter design must be equal to or greater than $\lceil \log_2(L+1) \rceil$.

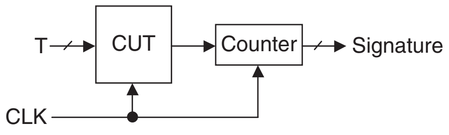


FIGURE 3.24

Ones counter as ORA.

3.4.2.2 *Transition count testing*

The theory behind transition count testing is similar to that for ones count testing, except the signature is defined as the number of 0-to-1 and 1-to-0 transitions. The **transition count test technique** [Hayes 1976] simply requires the use of a D flip-flop and an XOR gate connected to a ones counter (see Figure 3.25) to count the number of transitions in the output data stream. Consider the example given previously. Because $R_0 = \{0101100\}$, the signature or transition count of R_0, $TC(R_0)$, will be 4. Assume that the initial state of the D flip-flop, r_{-1}, is 0. Fault f_1 causing an erroneous response $R_1 = \{1100110\}$ will not be detected because $TC(R_1) = TC(R_0) = 4$, whereas fault f_2 causing $R_2 = \{0101010\}$ will be detected because $TC(R_2) = 6$.

Let the fault-free signature or transition count be m. Because a given L-bit sequence R_0 that starts with $r_0 = 0$ has $L - 1$ possible transitions, the number of sequences with m transitions can be given by $C(L - 1, m)$. Because R_0 can also start with $r_0 = 1$, there will be a total of $2C(L - 1, m)$ possible ways having m 0-to-1 and 1-to-0 transitions in an L-bit stream. Assuming all faulty sequences are equally likely to occur as the response of the CUT, the *aliasing probability* or *masking probability* of the use of transition count testing having m transitions [Savir 1985] is

$$P_{TC}(m) = \left(2C(L-1,m) - 1\right)/(2^L - 1)$$

In the previous example, where $m = TC(R_0) = 4$ and $L = 7$, $P_{TC}(m) = 29/127 = 0.23$. Figure 3.25 shows the transition count test circuit. The number of stages in the counter design must be equal to or greater than $\lceil \log_2(L + 1) \rceil$.

3.4.2.3 *Signature analysis*

Signature analysis is the most popular response compaction technique used today. The compaction scheme, based on **cyclic redundancy checking** (CRC) [Peterson 1972], was first developed in [Benowitz 1975]. Hewlett-Packard commercialized the first logic analyzer, called HP 5004A Signature Analyzer, based on the scheme and referred to it as **signature analysis** [Frohwerk 1977].

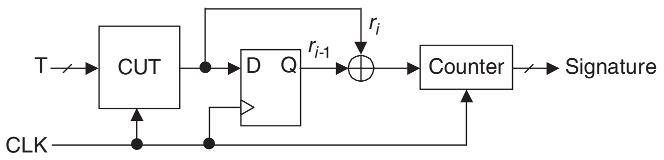


FIGURE 3.25

Transition counter as ORA.

In this subsection, we will discuss two signature analysis schemes: (1) **serial signature analysis** for compacting responses from a CUT having a single output and (2) **parallel signature analysis** for compacting responses from a CUT having multiple outputs.

3.4.2.3.1 Serial Signature Analysis

Consider the n-stage ***single-input signature register*** (SISR) shown in Figure 3.26. This SISR uses an additional XOR gate at the input for compacting an L-bit output sequence, M, into the *modular LFSR*. Let $M = \{m_0\, m_1\, m_2 \ldots m_{L-1}\}$, and define:

$$M(x) = m_0 + m_1x + m_2x^2 + \ldots + m_{L-1}x^{L-1}$$

After shifting the L-bit output sequence, M, into the *modular LFSR*, the contents (remainder) of the SISR, R, is given as $\{r_0\, r_1\, r_2 \ldots r_{n-1}\}$, or

$$r(x) = r_0 + r_1x + r_2x^2 + \ldots + r_{n-1}x^{n-1}$$

The SISR is basically a *CRC code generator* [Peterson 1972] or a *cyclic code checker* [Benowitz 1975]. Let the *characteristic polynomial* of the modular LFSR be $f(x)$. The authors in [Peterson 1972] have shown that the SISR performs polynomial division of $M(x)$ by $f(x)$, or

$$M(x) = q(x)f(x) + r(x)$$

The final state or **signature** in the SISR is the *polynomial remainder,* $r(x)$, of the division. Consider the four-stage SISR given in Figure 3.27 with $f(x) = 1 + x + x^4$. Assuming $M = \{10011011\}$, we can express $M(x) = 1 + x^3 + x^4 + x^6 + x^7$. By use of polynomial division, we obtain $q(x) = x^2 + x^3$ and $r(x) = 1 + x^2 + x^3$ or $R = \{1011\}$. The remainder $\{1011\}$ is equal to the *signature* derived from Figure 3.27a when the SISR is first initialized to a *starting pattern* (*seed*) of $\{0000\}$.

Now, assume fault f_1 produces an erroneous output stream $M' = \{11001011\}$ or $M'(x) = 1 + x + x^4 + x^6 + x^7$, as given in Figure 3.27b. By use of polynomial division, we obtain $q'(x) = x^2 + x^3$ and $r'(x) = 1 + x + x^2$ or $R' = \{1110\}$. Because the faulty signature R', $\{1110\}$, is different from the fault-free signature R, $\{1011\}$, fault f_1 is detected. For fault f_2 with $M'' = \{11001101\}$ or $M''(x) = 1 + x + x^4 + x^5 + x^7$ as given in Figure 3.27c, we have $q''(x) = x + x^3$ and $r''(x) = 1 + x^2 + x^3$ or $R'' = \{1011\}$. Because $R'' = R$, fault f_2 is not detected.

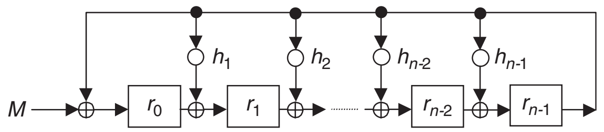
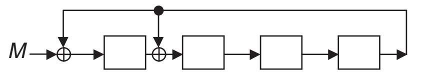


FIGURE 3.26

An n-stage single-input signature register (SISR).



M	r_0	r_1	r_2	r_3
1	**0**	**0**	**0**	**0**
1	1	0	0	0
0	1	1	0	0
1	0	1	1	0
1	1	0	1	1
0	0	0	0	1
0	1	1	0	0
1	0	1	1	0
R	**1**	**0**	**1**	**1**

(a)

M'	r_0	r_1	r_2	r_3
1	**0**	**0**	**0**	**0**
1	1	0	0	0
0	1	1	0	0
1	0	1	1	0
0	1	0	1	1
0	1	0	0	1
1	1	0	0	0
1	1	1	0	0
R'	**1**	**1**	**1**	**0**

(b)

M''	r_0	r_1	r_2	r_3
1	**0**	**0**	**0**	**0**
0	1	0	0	0
1	0	1	0	0
1	1	0	1	0
0	1	1	0	1
0	1	0	1	0
1	0	1	0	1
1	0	1	1	0
R''	**1**	**0**	**1**	**1**

(c)

FIGURE 3.27

A four-stage SISR: (a) Fault-free signature. (b) Signature for fault f_1. (c) Signature for fault f_2.

The *fault detection* or *aliasing* problem of an SISR can be better understood by looking at the *error sequence E* or *error polynomial E*(x) of the fault-free sequence M and a faulty sequence M'. Define $E = M + M'$, or:

$$E(x) = M(x) + M'(x)$$

If $E(x)$ is not divisible by $f(x)$, then all faults generating the faulty sequence M' will be detected. Otherwise, these faults are not detected. Consider fault f_1 again. We obtain $E = \{01010000\} = M + M' = \{10011011\} + \{11001011\}$ or $E(x) = x + x^3$. Because $E(x)$ is not divisible by $f(x) = 1 + x + x^4$, fault f_1 is detected. Consider fault f_2 again. We have $E = \{01010110\} = M + M'' = \{10011011\} + \{11001101\}$ or $E(x) = x + x^3 + x^5 + x^6$. Because $f(x)$ divides $E(x)$, *i.e.,* $E(x) = (x + x^2)f(x)$, fault f_2 is not detected.

Assume the SISR consists of n stages. For a given L-bit sequence, $L > n$, there are $2^{(L-n)}$ possible ways of producing an n-bit signature of which one is the correct signature. Because there are a total of $2^L - 1$ erroneous sequences in an L-bit stream, the *aliasing probability* with an n-stage SISR for *serial signature analysis* (SSA) is:

$$P_{SSA}(n) = \left(2^{(L-n)} - 1\right)/(2^L - 1)$$

If $L >> n$, then $P_{SSA}(n) \approx 2^{-n}$. When $n = 20$, $P_{SSA}(n) < 2^{-20} = 0.0001\%$.

3.4.2.3.2 Parallel Signature Analysis

A common problem when using ones count testing, transition count testing, and serial signature analysis is the excessive hardware cost required to test an m-output CUT. It is possible to reduce the hardware cost by use of an m-to-1 multiplexer, but this increases the test time m times.

Consider the n-stage ***multiple-input signature register*** (MISR) shown in Figure 3.28. The MISR uses n extra XOR gates for compacting n L-bit output sequences, M_0 *to* M_{n-1}, into the *modular LFSR* simultaneously.

[Hassan 1984] has shown that the n-input MISR can be remodeled as a single-input SISR with *effective input sequence* $M(x)$ and *effective error polynomial* $E(x)$ expressed as:

$$M(x) = M_0(x) + xM_1(x) + \ldots + x^{n-2}M_{n-2}(x) + x^{n-1}M_{n-1}(x)$$

and

$$E(x) = E_0(x) + xE_1(x) + \ldots + x^{n-2}E_{n-2}(x) + x^{n-1}E_{n-1}(x)$$

Consider the four-stage MISR shown in Figure 3.29 that uses $f(x) = 1 + x + x^4$. Let $M_0 = \{10010\}$, $M_1 = \{01010\}$, $M_2 = \{11000\}$, and $M_3 = \{10011\}$. From this information, the signature R of the MISR can be calculated as {1011}. With $M(x) = M_0(x) + xM_1(x) + x^2M_2(x) + x^3M_3(x)$, we obtain $M(x) = 1 + x^3 + x^4 + x^6 + x^7$ or $M = \{10011011\}$ as shown in Figure 3.30. This is the same data stream we used in the SISR example in Figure 3.27a. Therefore, $R = \{1011\}$.

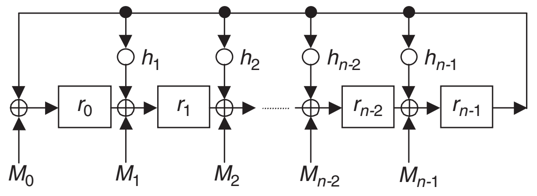


FIGURE 3.28

An n-stage multiple-input signature register (MISR).

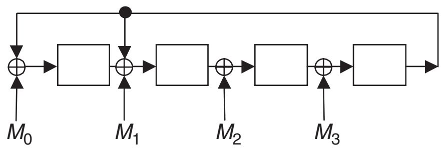


FIGURE 3.29

A four-stage MISR.

M_0	1	0	0	1	0			
M_1		0	1	0	1	0		
M_2			1	1	0	0	0	
M_3				1	0	0	1	1
M	1	0	0	1	1	0	1	1

FIGURE 3.30

An equivalent M sequence.

Assume there are m L-bit sequences to be compacted in an n-stage MISR, where $L > n \geq m \geq 2$. The *aliasing probability* for *parallel signature analysis* (PSA) now becomes:

$$P_{PSA}(n) = \left(2^{(mL-n)} - 1\right)/(2^{mL} - 1)$$

If $L >> n$, *then* $P_{PSA}(n) \approx 2^{-n}$. When $n = 20$, $P_{PSA}(n) < 2^{-20} = 0.0001\%$. The result suggests that $P_{PSA}(n)$ mainly depends on n, when $L >> n$. Hence, increasing the number of MISR stages or the use of the same MISR but with a different $f(x)$ can substantially reduce the *aliasing probability* [Hassan 1984; Williams 1987].

3.4.3 Logic BIST architectures

Several architectures for incorporating **offline BIST** techniques into a design have been proposed. These BIST architectures can be classified into two classes: (1) those that use the **test-per-scan BIST** scheme and (2) those that use the **test-per-clock BIST** scheme. The *test-per-scan BIST* scheme takes advantage of the already built-in scan chains of the scan design and applies a test pattern to the CUT after a shift operation is completed; hence, the hardware overhead is low. The *test-per-clock BIST* scheme, however, applies a test pattern to the CUT and captures its test response every system clock cycle; hence, the scheme can execute tests much faster than the test-per-scan BIST scheme but at an expense of more hardware overhead.

In this subsection, we only discuss three representative BIST architectures, the first two for pseudo-random testing and the last for pseudo-exhaustive testing. Although pseudo-random testing is commonly adopted in industry, the exhaustive and pseudo-exhaustive test techniques are applicable for designs that use the test-per-clock BIST scheme. For a more comprehensive survey of these BIST architectures, refer to [Abramovici 1994; Bardell 1987; McCluskey 1985; Wang 2006a]. Fault coverage enhancement with the pseudo-random test technique can also be found in [Tsai 1999; Wang 2006a; Lai 2007].

3.4.3.1 *Self-testing with MISR and parallel SRSG (STUMPS)*

A test-per-scan BIST design was presented in [Bardell 1982]. This design, shown in Figure 3.31, contains a PRPG (parallel ***shift register sequence generator*** [SRSG]) and a MISR. The scan chains are loaded in parallel from the PRPG. The system clocks are then triggered, and the test responses are shifted to the MISR for compaction. New test patterns are shifted in at the same time while test responses are being shifted out. This BIST architecture that uses the test-per-scan BIST scheme is referred to as ***self-testing with MISR and parallel SRSG*** **(STUMPS)** [Bardell 1982].

Because of the ease of integration with traditional scan architecture, the **STUMPS** architecture is the only BIST architecture widely used in industry to

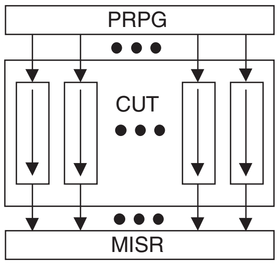


FIGURE 3.31

STUMPS.

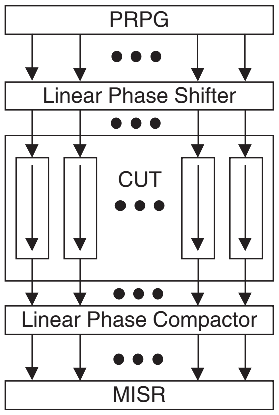


FIGURE 3.32

A STUMPS-based architecture.

date. To further reduce the lengths of the PRPG and MISR and improve the randomness of the PRPG, a STUMPS-based architecture that includes an optional linear phase shifter and an optional linear phase compactor is often used in industrial applications [Nadeau-Dostie 2000; Cheon 2005]. The linear phase shifter and linear phase compactor typically comprise a network of XOR gates. Figure 3.32 shows the STUMPS-based architecture.

3.4.3.2 *Built-in logic block observer (BILBO)*

The architecture described in [Könemann 1979, 1980] applies to circuits that can be partitioned into independent modules (logic blocks). Each module is assumed to have its own input and output registers (storage elements), or such registers are added to the circuit where necessary. The registers are redesigned so that for test purposes they act as PRPGs for test generation or MISRs for signature analysis. The redesigned register is called a ***built-in logic block observer*** (BILBO).

The BILBO is operated in four modes: normal mode, scan mode, test generation or signature analysis mode, and reset mode. A typical three-stage BILBO, which is reconfigurable into a TPG or a MISR during self-test is shown in Figure 3.33. It is controlled by two control inputs B_1 and B_2. When both control inputs B_1 and B_2 are equal to 1, the circuit functions in normal mode with the inputs Y_i gated directly into the D flip-flops. When both control inputs are equal to 0, the BILBO is configured as a shift register. Test data can be shifted in through the serial scan-in port or shifted out through the serial scan-out port. Setting $B_1 = 1$ and $B_2 = 0$ converts the BILBO into a MISR. It can then be used in this configuration as a TPG by holding every Y_i input to 1. The BILBO is reset after a system clock is triggered when $B_1 = 0$ and $B_2 = 1$.

This technique is most suitable for testing circuits, such as *random-access memories* (RAMs), *read-only memories* (ROMs), or bus-oriented circuits, where input and output registers of the partitioned modules can be reconfigured independently. For testing finite-state machines or pipeline-oriented circuits as shown in Figure 3.34, the signature data from the previous module must be

B_1	B_2	**Operation mode**
1	1	Normal
0	0	Scan
1	0	Mixed Test Generation and Signature Analysis
0	1	Reset

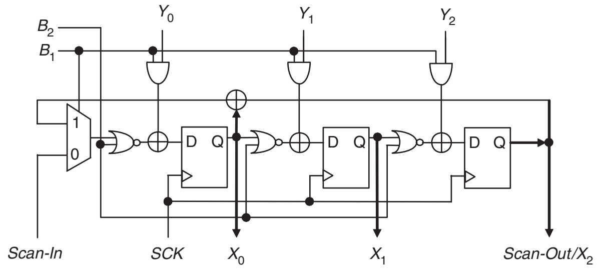


FIGURE 3.33

A three-stage built-in logic block observer (BILBO).

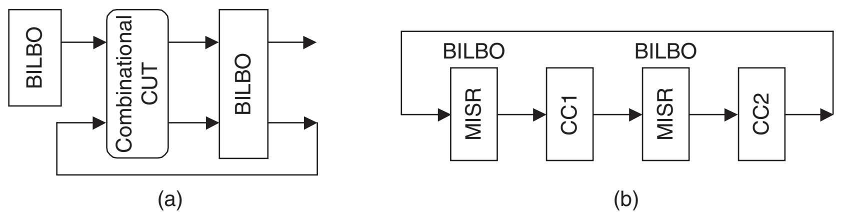


FIGURE 3.34

BILBO architectures: (a) For testing a finite-state machine. (b) For testing a pipeline-oriented circuit.

used as test patterns for the next module, because the test generation and signature analysis modes cannot be separated. In this case, a detailed fault simulation is required to achieve 100% single-stuck fault coverage.

3.4.3.3 *Concurrent built-in logic block observer (CBILBO)*

One technique to overcome the above BILBO fault coverage loss problem is to use the ***concurrent built-in logic block observer*** (CBILBO) approach [Wang 1986b]. Reconfigured from the BILBO design, the CBILBO is based on the test-per-clock BIST scheme and uses two registers to perform test generation and signature analysis simultaneously. A CBILBO design is illustrated in Figure 3.35, where only three modes of operation are considered: normal, scan, and test generation and signature analysis. When $B_1 = 0$ and $B_2 = 1$, the upper D flip-flops act as a MISR for signature analysis, whereas the lower two-port D flip-flops form a TPG for test generation. Because signature analysis is separated from test generation, an *exhaustive* or *pseudo-exhaustive pattern generator* (EPG/PEPG) can now be used for test generation; therefore, no fault simulation is required, and it is possible to achieve 100% single-stuck fault coverage with the CBILBO architectures for testing designs shown in Figure 3.36. However, the hardware cost associated with the use of the CBILBO approach is generally higher than for the STUMPS approach.

3.4.4 Industry practices

Logic BIST has a history of more than 30 years since its invention in the 1970s. Although it is only a few years behind the invention of scan, logic BIST has yet

B_1	B_2	**Operation mode**
–	0	Normal
1	1	Scan
0	1	Test Generation and Signature Analysis

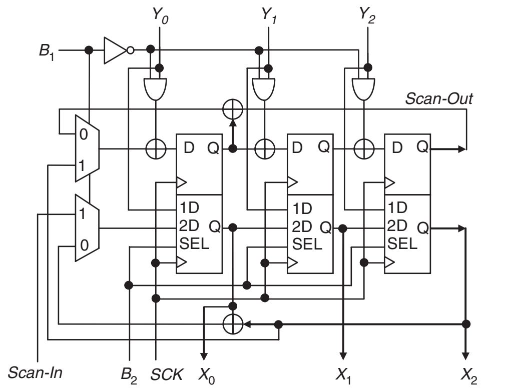


FIGURE 3.35

A three-stage concurrent BILBO (CBILBO).

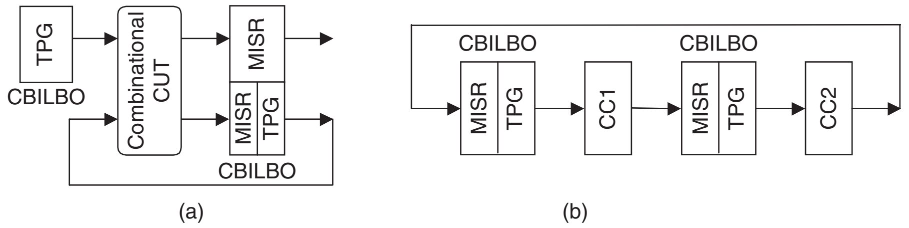


FIGURE 3.36

CBILBO architectures: (a) For testing a finite-state machine. (b) For testing a pipeline-oriented circuit.

to gain strong industry support. The worldwide market is estimated to be close to 10% of the scan market. The logic BIST products available in the marketplace now include **Encounter Test** from Cadence Design Systems [Cadence 2008], **ETLogic** from LogicVision [LogicVision 2008], **LBIST Architect** from Mentor Graphics [Mentor 2008], and **TurboBIST-Logic** from SynTest Technologies [SynTest 2008]. The logic BIST product offered in Encounter Test by Cadence currently includes support for test structure extraction, verification, logic simulation for signatures, and fault simulation for coverage. Unlike all other three BIST vendors that provide their own logic BIST structures in their respective products, Cadence offers a service to insert custom logic BIST structures or to use any customer-inserted logic BIST structures, including working with the customer to have custom on-chip clocking for logic BIST. A similar case exists in ETLogic from LogicVision when the double-capture clocking scheme is used.

All these commercially available logic BIST products support the STUMPS-based architectures. Cadence supports a weighted-random spreading network (XOR network) for STUMPS with multiple-weight selects [Foote 1997]. For at-speed delay fault testing, ETLogic [LogicVision 2008] uses a **skewed-load-based at-speed BIST architecture**; TurboBIST-Logic [Wang 2005b, 2006b; SynTest 2008] implements the **double-capture-based at-speed BIST architecture**; and LBIST Architect [Mentor 2008] adopts a **hybrid at-speed BIST architecture** that supports both skewed-load and double-capture. In addition, all products provide inter-clock-domain delay fault testing for synchronous clock domains. On-chip clock controllers for testing these inter-clock-domain faults at-speed can be found in [Rajski 2003; Furukawa 2006; Nadeau-Dostie 2006, 2007; Keller 2007], and Table 3.7 summarizes the capture-clocking schemes for at-speed logic BIST that is used by the EDA vendors.

3.5 TEST COMPRESSION

Test compression can provide 10× to 100× reduction or even more in the amount of test data (both test stimulus and test response) that must be stored on the ***automatic test equipment*** (ATE) [Touba 2006; Wang 2006a] for testing

Table 3.7 Summary of Industry Practices for At-Speed Logic BIST

Industry Practices	Skewed-Load	Double-Capture
Encounter test	Through service	Through service
ETLogic	√	Through service
LBIST Architect	√	√
TurboBIST-Logic		√

with a deterministic ATPG-generated test set. This greatly reduces ATE memory requirements and even more importantly reduces test time, because less data have to be transferred across the limited bandwidth between the ATE and the chip. Moreover, test compression methods are easy to adopt in industry because they are compatible with the conventional design rules and test generation flows used for scan testing.

Test compression is achieved by adding some additional on-chip hardware before the scan chains to decompress the test stimulus coming from the tester and after the scan chains to compact the response going to the tester. This is illustrated in Figure 3.37. This extra on-chip hardware allows the test data to be stored on the tester in a compressed form. Test data are inherently highly compressible because typically only 1% to 5% of the bits on a test pattern that is generated by an ATPG program have specified (*care*) values. Lossless compression techniques can thus be used to significantly reduce the amount of test stimulus data that must be stored on the tester. The on-chip **decompressor** expands the compressed test stimulus back into the original test patterns (matching in all the care bits) as they are shifted into the scan chains. The on-chip **compactor** converts long output response sequences into short signatures. Because the compaction is lossy, some fault coverage can be lost because

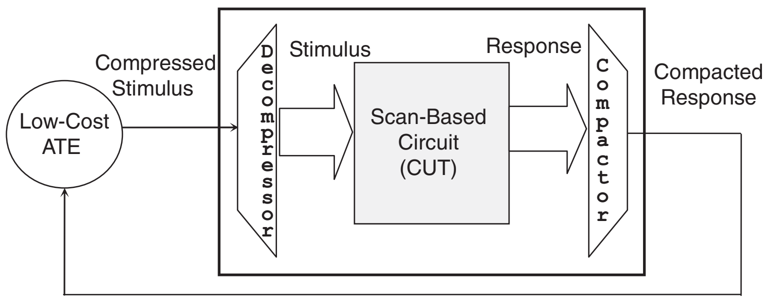


FIGURE 3.37

Architecture for test compression.

of unknown (X) values that might appear in the output sequence or aliasing where a faulty output response signature is identical to the fault-free output response signature. With proper design of the ***circuit under test*** (CUT) and the compaction circuitry, however, the fault coverage loss can be kept negligibly small.

3.5.1 Circuits for test stimulus compression

A **test cube** is defined as a deterministic test vector in which the bits that are not assigned values by the ATPG procedure are left as don't cares (X's). Normally, ATPG procedures perform *random fill* in which all the X's in the test cubes are filled randomly with 1's and 0's to create fully specified test vectors; however, for test stimulus compression, random fill is not performed during ATPG so the resulting test set consists of incompletely specified test cubes. The X's make the test cubes much easier to compress than fully specified test vectors.

As mentioned earlier, test stimulus compression should be an information lossless procedure with respect to the specified (care) bits to preserve the fault coverage of the original test cubes. After decompression, the resulting test patterns shifted into the scan chains should match the original test cubes in all the specified (care) bits.

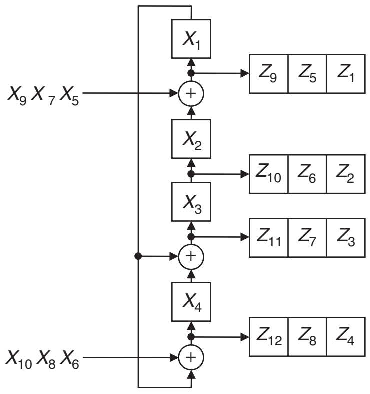
Many schemes for compressing test cubes have been surveyed in [Touba 2006; Wang 2006a]. Two schemes based on linear decompression and broadcast scan are described here in greater detail mainly because the industry has favored both approaches over code-based schemes from area overhead and compression ratio points of view. These code-based schemes can be found in [Wang 2006a].

3.5.1.1 *Linear-decompression-based schemes*

A class of test stimulus compression schemes is based on the use of **linear decompressors** to expand the data coming from the tester to fill the scan chains. Any decompressor that consists of only XOR gates and flip-flops is a ***linear decompressor*** [Könemann 1991]. Linear decompressors have a very useful property: their *output space* (*i.e.*, the space of all possible test vectors that they can generate) is a linear subspace that is spanned by a Boolean matrix. In other words, for any linear decompressor that expands an m-bit compressed stimulus from the tester into an n-bit stimulus (test vector), there exists a Boolean matrix $\boldsymbol{A}_{n\times m}$ such that the set of test vectors that can be generated by the linear decompressor is spanned by $\boldsymbol{A}$. A test vector $\boldsymbol{Z}$ can be compressed by a particular linear decompressor if and only if there exists a solution to a system of linear equations, $\boldsymbol{AX} = \boldsymbol{Z}$, where $\boldsymbol{A}$ is the **characteristic matrix** of the linear decompressor and X is a set of **free variables** stored on the tester (every bit stored on the tester can be thought of as a "free variable" that can be assigned any value, 0 or 1).

The characteristic matrix for a linear decompressor can be obtained by symbolic simulation where each free variable coming from the tester is represented by a symbol. An example of this is shown in Figure 3.38, where a sequential linear decompressor containing an LFSR is used. The initial state of the LFSR is represented by free variables X_1 to X_4, and the free variables X_5 to X_{10} are shifted in from two channels as the scan chains are loaded. After symbolic simulation, the final values in the scan chains are represented by the equations for Z_1 *to* Z_{12}. The corresponding system of linear equations for this linear decompressor is shown in Figure 3.39.

The symbolic simulation goes as follows. Assume that the initial seed X_1 *to* X_4 has been already loaded into the flip-flops. In the first clock cycle, the top flip-flop is loaded with the XOR of X_2 and X_5; the second flip-flop is loaded with X_3; the third flip-flop is loaded with the XOR of X_1 and X_4; and the bottom flip-flop is loaded with the XOR of X_1 and X_6. Thus, we obtain $Z_1 = X_2 \oplus X_5$, $Z_2 = X_3$, $Z_3 = X_1 \oplus X_4$, and $Z_4 = X_1 \oplus X_6$. In the second clock cycle, the top flip-flop is loaded with the XOR of the contents of the second flip-flop (X_3) and X_7; the second flip-flop is loaded with the contents of the third flip-flop ($X_1 \oplus X_4$); the third flip-flop is loaded with the XOR of the contents of the first flip-flop ($X_2 \oplus X_5$) and the fourth flip-flop ($X_1 \oplus X_6$); and the bottom flip-flop is loaded with the XOR of the contents of the first flip-flop ($X_2 \oplus X_5$) and X_8. Thus, we obtain $Z_5 = X_3 \oplus X_7$, $Z_6 = X_1 \oplus X_4$, $Z_7 = X_1 \oplus X_2 \oplus X_5 \oplus X_6$, and $Z_8 = X_2 \oplus X_5 \oplus X_8$. In the third clock cycle, the top flip-flop is loaded with



$Z_9 = X_1 \oplus X_4 \oplus X_9$	$Z_5 = X_3 \oplus X_7$	$Z_1 = X_2 \oplus X_5$
$Z_{10} = X_1 \oplus X_2 \oplus X_5 \oplus X_6$	$Z_6 = X_1 \oplus X_4$	$Z_2 = X_3$
$Z_{11} = X_2 \oplus X_3 \oplus X_5 \oplus X_7 \oplus X_8$	$Z_7 = X_1 \oplus X_2 \oplus X_5 \oplus X_6$	$Z_3 = X_1 \oplus X_4$
$Z_{12} = X_3 \oplus X_7 \oplus X_{10}$	$Z_8 = X_2 \oplus X_5 \oplus X_8$	$Z_4 = X_1 \oplus X_6$

FIGURE 3.38

Example of symbolic simulation for linear decompressor.

$$\begin{pmatrix} 0&1&0&0&1&0&0&0&0&0\\ 0&0&1&0&0&0&0&0&0&0\\ 1&0&0&1&0&0&0&0&0&0\\ 1&0&0&0&0&1&0&0&0&0\\ 0&0&1&0&0&0&1&0&0&0\\ 1&0&0&1&0&0&0&0&0&0\\ 1&1&0&0&1&1&0&0&0&0\\ 0&1&0&0&1&0&0&1&0&0\\ 1&0&0&1&0&0&0&0&1&0\\ 1&1&0&0&1&1&0&0&0&0\\ 0&1&1&0&1&0&1&1&0&0\\ 0&0&1&0&0&0&1&0&0&1 \end{pmatrix} \begin{pmatrix} X_1\\ X_2\\ X_3\\ X_4\\ X_5\\ X_6\\ X_7\\ X_8\\ X_9\\ X_{10} \end{pmatrix} = \begin{pmatrix} Z_1\\ Z_2\\ Z_3\\ Z_4\\ Z_5\\ Z_6\\ Z_7\\ Z_8\\ Z_9\\ Z_{10}\\ Z_{11}\\ Z_{12} \end{pmatrix}$$

FIGURE 3.39

System of linear equations for the decompressor in Figure 3.38.

the XOR of the contents of the second flip-flop ($X_1 \oplus X_4$) and X_9; the second flip-flop is loaded with the contents of the third flip-flop ($X_1 \oplus X_2 \oplus X_5 \oplus X_6$); the third flip-flop is loaded with the XOR of the contents of the first flip-flop ($X_3 \oplus X_7$) and the fourth flip-flop ($X_2 \oplus X_5 \oplus X_8$); and the bottom flip-flop is loaded with the XOR of the contents of the first flip-flop ($X_3 \oplus X_7$) and X_{10}. Thus, we obtain $Z_9 = X_4 \oplus X_9$, $Z_{10} = X_1 \oplus X_6$, $Z_{11} = X_2 \oplus X_5 \oplus X_8$, and $Z_{12} = X_3 \oplus X_7 \oplus X_{10}$. At this point, the scan chains are fully loaded with a test cube, so the simulation is complete.

3.5.1.1.1 Combinational linear decompressors

The simplest linear decompressors use only combinational XOR networks. Each scan chain is fed by the XOR of some subset of the channels coming from the tester [Bayraktaroglu 2001, 2003; Könemann 2003; Mitra 2006; Han 2007; Wang 2004, 2008]. The advantage compared with sequential linear decompressors is simpler hardware and control. The drawback is that, to encode a test cube, each **scan slice** (the *n*-bits that are loaded into the *n* scan chains in each clock cycle) must be encoded with only the free variables that are shifted from the tester in a single clock cycle (which is equal to the number of channels). The worst-case most highly specified scan slices tend to limit the amount of compression that can be achieved, because the number of channels from the tester has to be sufficiently large to encode the most highly specified scan slices. Consequently, it is very difficult to obtain a high **encoding efficiency** (typically it will be less than 0.25); for the other less specified scan slices, a lot of the free variables end up getting wasted, because those scan slices could have been encoded with many fewer free variables.

One approach for improving the encoding efficiency of combinational linear decompressors that was proposed in [Krishna 2003] is to dynamically adjust the number of scan chains that are loaded in each clock cycle. So for a highly specified scan slice, four clock cycles could be used in which 25% of the scan chains are loaded in each cycle, whereas for a lightly specified scan slice, only one clock cycle can be used in which 100% of the scan slices are loaded. This allows a better matching of the number of free variables with the number of specified bits to achieve a higher encoding efficiency. Note that it requires that the scan clock be divided into multiple domains.

3.5.1.1.2 Sequential linear decompressors

Sequential linear decompressors are based on linear finite-state machines such as LFSRs, cellular automata, or ring generators [Mrugalski 2004]. The advantage of a sequential linear decompressor is that it allows free variables from earlier clock cycles to be used when encoding a scan slice in the current clock cycle. This provides much greater flexibility than combinational decompressors and helps avoid the problem of the worst-case most highly specified scan slices limiting the overall compression. The more flip-flops that are used in the sequential linear decompressor, the greater the flexibility that is provided. [Tobua 2006] classifies the sequential linear decompressors into two classes:

1. **Static reseeding** that computes a seed (an initial state) for each test cube [Touba 2006]. This seed, when loaded into an LFSR and run in autonomous mode, will produce the test cube in the scan chains [Könemann 1991]. This technique achieves compression by storing only the seeds instead of the full test cubes.
2. **Dynamic reseeding** calls for the injection of free variables coming from the tester into the LFSR as it loads the scan chains [Krishna 2001; Könemann 2001; Rajski 2004].

Figure 3.40 shows a generic example of a sequential linear decompressor that uses *b* channels from the tester to continuously inject free variables into the LFSR as it loads the scan chains through a combinational linear decompressor that typically is a combinational XOR network.

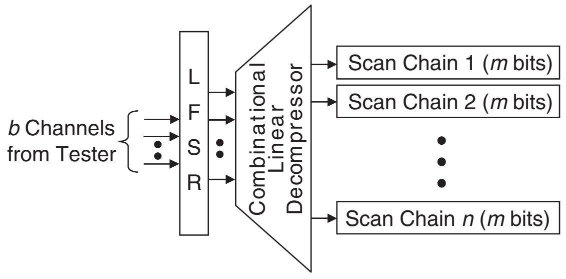


FIGURE 3.40

Typical sequential linear decompressor.

3.5.1.2 ***Broadcast-scan-based schemes***

Another class of test stimulus compression schemes is based on broadcasting the same value to multiple scan chains. This was first proposed in [Lee 1998] and [Lee 1999]. Because of its simplicity and effectiveness, this method has been used as the basis of many test compression architectures, including some commercial *design for testability* (DFT) tools.

3.5.1.2.1 Broadcast scan

To illustrate the basic concept of **broadcast scan**, first consider two independent circuits C_1 and C_2. Assume that these two circuits have their own test sets $T_1 = < t_{11}, t_{12}, \ldots, t_{1k} >$ and $T_2 = < t_{21}, t_{22}, \ldots, t_{2l} >$, respectively. In general, a test set may consist of random patterns and deterministic patterns. In the beginning of the ATPG process, usually random patterns are initially used to detect the easy-to-detect faults. If the same random patterns are used when generating both T_1 and T_2, then we may have $t_{11} = t_{21}$, $t_{12} = t_{22}, \ldots$, up to some ith pattern. After most faults have been detected by the random patterns, deterministic patterns are generated for the remaining difficult-to-detect faults. Generally, these patterns have many "don't care" bits. For example, when generating $t_{1(i+1)}$, many "don't care" bits may still exist when no more faults in C_1 can be detected. By use of a test pattern with bits assigned so far for C_1, we can further assign specific values to the "don't care" bits in the pattern to detect faults in C_2. Thus, the final pattern would be effective in detecting faults in both C_1 and C_2.

The concept of pattern sharing can be extended to multiple circuits as illustrated in Figure 3.41. One major advantage of the use of *broadcast scan* for independent circuits is that all faults that are detectable in all original circuits will also be detectable with the broadcast structure. This is because if one test vector can detect a fault in a stand-alone circuit, then it will still be possible to apply this vector to detect the fault in the broadcast structure. Thus, the broadcast scan method will not affect the fault coverage if all circuits are independent. Note that broadcast scan can also be applied to multiple scan chains of a single circuit if all subcircuits driven by the scan chains are independent.

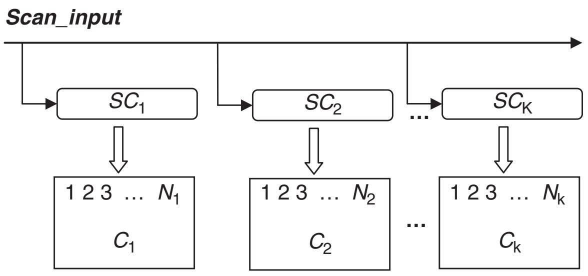


FIGURE 3.41

Broadcasting to scan chains driving independent circuits.

3.5.1.2.2 Illinois scan

If *broadcast scan* is used for multiple scan chains of a single circuit where the subcircuits driven by the scan chains are not independent, then the property of always being able to detect all faults is lost. The reason for this is that if two scan chains are sharing the same channel, then the *i*th scan cell in each of the two scan chains will always be loaded with identical values. If some fault requires two such scan cells to have opposite values to be detected, it will not be possible to detect this fault with broadcast scan.

To address the problem of some faults not being detected when broadcast scan is used for multiple scan chains of a single circuit, the **Illinois scan architecture** was proposed in [Hamzaoglu 1999] and [Hsu 2001]. This scan architecture consists of two modes of operations, namely a *broadcast mode* and a *serial scan mode,* which are illustrated in Figure 3.42. The *broadcast mode* is first used to detect most faults in the circuit. During this mode, a scan chain is divided into multiple subchains called *segments,* and the same vector can be shifted into all segments through a single shared scan-in input. The response data from all subchains are then compacted by a MISR or other space/time compactor. For the remaining faults that cannot be detected in broadcast mode, the *serial scan mode* is used where any possible test pattern can be applied. This ensures that complete fault coverage can be achieved. The extra logic required to implement the Illinois scan architecture consists of several multiplexers and some simple control logic to switch between the two modes. The area overhead of this logic is typically quite small compared with the overall chip area.

The main drawback of the Illinois scan architecture is that no test compression is achieved when it is run in *serial scan mode*. This can significantly degrade the overall **compression ratio** if many test patterns must be applied in serial scan mode. To reduce the number of patterns that need to be applied in serial scan mode, multiple-input broadcast scan or reconfigurable broadcast scan can be used. These techniques are described next.

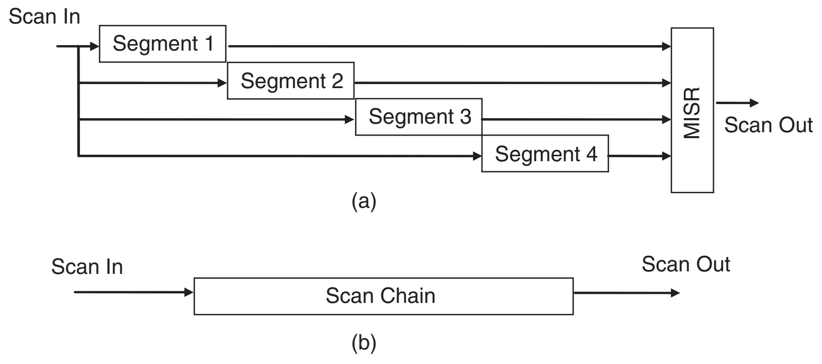


FIGURE 3.42

Two modes of Illinois scan architecture: (a) Broadcast mode. (b) Serial scan mode.

3.5.1.2.3 Multiple-input broadcast scan

Instead of the use of only one channel to drive all scan chains, a **multiple-input broadcast scan** could be used where there is more than one channel [Shah 2004]. Each channel can drive some subset of the scan chains. If two scan chains must be independently controlled to detect a fault, then they could be assigned to different channels. The more channels that are used and the shorter each scan chain is, the easier to detect more faults because fewer constraints are placed on the ATPG. Determining a configuration that requires the minimum number of channels to detect all detectable faults is thus highly desired with a multiple-input broadcast scan technique.

3.5.1.2.4 Reconfigurable broadcast scan

Multiple-input broadcast scan may require a large number of channels to achieve high fault coverage. To reduce the number of channels that are required, a **reconfigurable broadcast scan** method can be used. The idea is to provide the capability to reconfigure the set of scan chains that each channel drives. Two possible reconfiguration schemes have been proposed, namely **static reconfiguration** [Pandey 2002; Wang 2002; Samaranayake 2003; Chandra 2007], and **dynamic reconfiguration** [Li 2004; Sitchinava 2004; Wang 2004, 2008; Mitra 2006; Wohl 2007a]. In *static reconfiguration*, the reconfiguration can only be done when a new pattern is to be applied. For this method, the target fault set can be divided into several subsets, and each subset can be tested by a single configuration. After testing one subset of faults, the configuration can be changed to test another subset of faults. In *dynamic reconfiguration*, the configuration can be changed while scanning in a pattern. This provides more reconfiguration flexibility and hence can, in general, lead to better results with fewer channels. This is especially important for hard cores, when the test patterns provided by core vendor cannot be regenerated. The drawback of dynamic reconfiguration *versus* static reconfiguration is that more control information is needed for reconfiguring at the right time, whereas for static reconfiguration the control information is much less because the reconfiguration is done only a few times (only after all the test patterns that use a particular configuration have been applied).

Figure 3.43 shows an example *multiplexer* (MUX) network that can be used for dynamic configuration. When a value on the control line is selected, particular data at the four input pins are broadcasted to the eight scan chain inputs. For instance, when the control line is set to 0 (or 1), the scan chain 1 output will receive input data from Pin 4 (or Pin 1) directly.

3.5.1.2.5 Virtual scan

Rather than the use of MUX networks for test stimulus compression, combinational logic networks can also be used as decompressors. The combinational logic network can consist of any combination of simple combinational gates, such as buffers, inverters, AND/OR gates, MUXs, and XOR gates. This scheme, referred to as **virtual scan**, is different from *reconfigurable broadcast scan* and

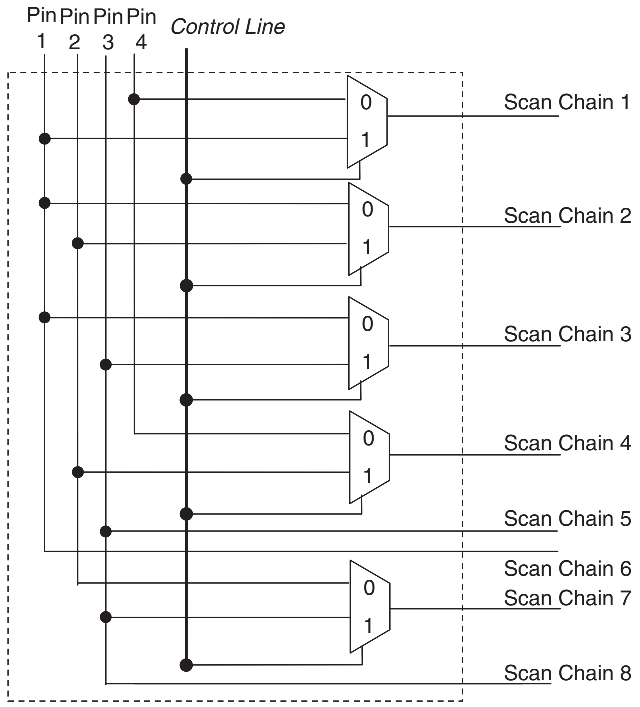


FIGURE 3.43

Example MUX network with control line(s) connected only to select pins of the multiplexers.

combinational linear decompression where pure MUX and XOR networks are allowed, respectively. The combinational logic network and the order of the scan chains can be specified as a set of constraints or just as an expanded circuit for ATPG. In either case, the test cubes that ATPG generates are the compressed stimuli for the decompressor itself. There is no need to solve a system of linear equations, and *dynamic compaction* can be effectively used during the ATPG process. Hence, only one-pass ATPG is required during test stimulus compression.

The *virtual scan* scheme was proposed in [Wang 2002, 2004, 2008]. In these papers, the decompressor was referred to as a **broadcaster**. The authors also proposed adding additional logic, when required, through *VirtualScan inputs* to reduce or remove the constraints imposed by the broadcaster on the circuit, thereby yielding very little or no fault coverage loss caused by test stimulus compression. For instance, a **scan connector** consisting of a set of multiplexers that places scan cells in the scan chains in a particular order can be connected to the outputs of the combinational logic network during each virtual scan test mode. Because the scan chains are reordered in each test mode, the imposed constraints of the combinational logic network on the circuit are reduced or removed.

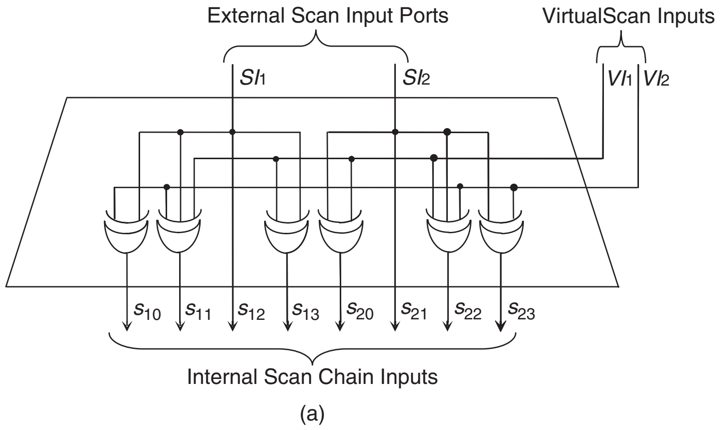
In a broad sense, *virtual scan* is a generalized class of broadcast scan, Illinois scan, multiple-input broadcast scan, reconfigurable broadcast scan, and combinational linear decompression. The advantage of the use of virtual scan is that it allows the ATPG to directly search for a test cube that can be applied by the decompressor and allows very effective dynamic compaction. Thus, virtual scan may produce shorter test sets than any test stimulus compression scheme based on solving linear equations; however, because this scheme may impose XOR or MUX constraints directly on the original circuit, it may take longer than those based on solving linear equations to generate test cubes or compressed stimuli. Two example virtual scan decompression circuits are shown in Figures 3.44a and 3.44b, respectively [Wang 2008]. Additional VirtualScan inputs are used to further reduce the XOR or MUX constraints imposed on the original circuit. An XOR network similar to the broadcaster shown in Figure 3.44a is sometimes referred to as a **space expander** or a **spreading network** in logic BIST applications.

3.5.2 **Circuits for test response compaction**

Test response compaction is performed at the outputs of the scan chains. The purpose is to reduce the amount of test response that needs to be transferred back to the tester. Although test stimulus compression must be lossless, test response compaction can be lossy. A large number of different test response compaction schemes and associated (response) compactors have been presented in the literature [Wang 2006a]. The effectiveness of each compaction scheme and the chosen compactor depends on its ability to avoid *aliasing* and tolerate *unknown test response bits* or *X*'s. These schemes can be grouped into three categories: (1) **space compaction**, (2) **time compaction**, and (3) **mixed space and time compaction**.

A **space compactor** compacts an m-bit-wide output pattern to an n-bit-wide output pattern (where $n < m$). A **time compactor** compacts p output patterns to q output patterns (where $q < p$). A **mixed space and time compactor** has both space and time compaction performed concurrently. Typically, a space compactor is composed of XOR gates [Saluja 1983]; a time compactor includes a *multiple-input signature register* (MISR) [Frohwerk 1977]; and a mixed space and time compactor adds a space compactor at either the input or the output side of a time compactor [Saluja 1983; Wohl 2001]. Because test response compaction can be combinational-logic-based or sequential-logic-based, without loss of generality, we refer space compaction to as a **combinational compaction** scheme, and time compaction as well as mixed space and time compaction to as **sequential compaction** schemes.

There are three sources of aliasing according to [Wohl 2001]: (1) **combinational cancellation** occurs when two or more erroneous scan chain outputs (compactor inputs) are XORed in the compactor during the same cycle, which


External Scan Input Ports
VirtualScan Inputs
SI_1
SI_2
VI_1
VI_2
s_{10} s_{11} s_{12} s_{13} s_{20} s_{21} s_{22} s_{23}
Internal Scan Chain Inputs
(a)

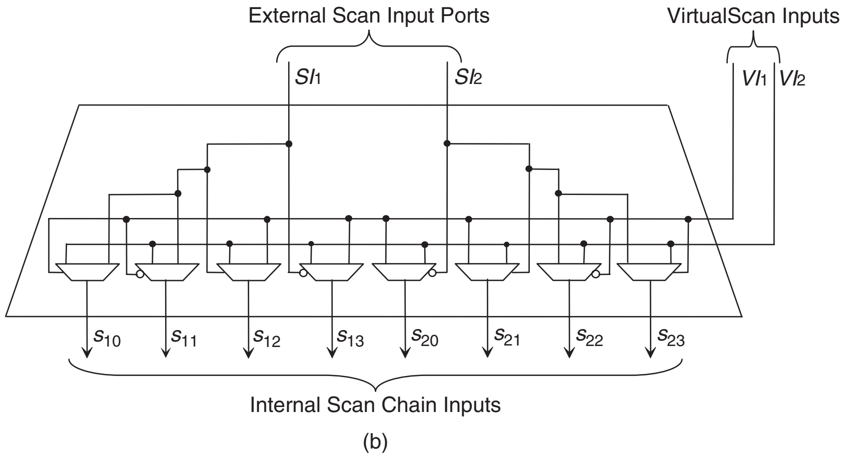

External Scan Input Ports
VirtualScan Inputs
SI_1
SI_2
VI_1
VI_2
s_{10} s_{11} s_{12} s_{13} s_{20} s_{21} s_{22} s_{23}
Internal Scan Chain Inputs
(b)

FIGURE 3.44

Example virtual scan decompression circuits: (a) Broadcaster that sees an example XOR network with additional VirtualScan inputs to reduce coverage loss. (b) Broadcaster that uses an example MUX network with additional VirtualScan inputs that can be also connected to data pins of the multiplexers.

cancel out the error effects in that cycle; (2) **shift cancellation** occurs when one or more erroneous scan chain output bits captured into the compactor are cancelled out by other erroneous scan chain output bits when the former are shifted down the shift path of the compactor; and (3) **feedback cancellation** occurs when one or more errors captured into the compactor during one cycle propagate through some feedback path of the compactor and cancel out with errors in later cycles. Combinational cancellation will exist in space compaction as well as mixed space and time compaction, because *non-aliasing* *space compactors* are impractical for real designs [Chakrabarty 1998; Pouya 1998]. On the other hand, shift cancellation and feedback cancellation are only present when either time compaction or mixed space and time compaction is used; however, shift cancellation is independent of the compactor feedback structure and its polynomial, whereas feedback cancellation depends on the compactor polynomial chosen.

Because unknown test response bits (*X*'s) can potentially reduce the fault coverage of the circuit under test when a combinational compactor is used and corrupt the final signature in a sequential compactor, one safe approach is to completely block these *X*'s before they reach the **response compactor** (combinational compactor or sequential compactor). During design, these potential **X-generators (X-sources)** can be identified with a scan design rule checker. When the *X* effects of an X-generator are likely to reach the response compactor, these *X*'s must be blocked before they reach the compactor [Gu 2001]. The process is often referred to as **X-blocking** or **X-bounding**.

In X-blocking, an X-source can be blocked either at the X-source or anywhere along its propagation paths before *X*'s reach the compactor. In case the X-source has been blocked at a nearby location during test and will not reach the compactor, there is no need to block the X-source; however, care must be taken to ensure that no observation points are added between the X-source and the location at which it is blocked to avoid capturing potential *X*'s into the compactor.

A simple example illustrating the X-blocking scheme for an X-source is shown in Figure 3.45. The output of the X-source is blocked and forced to 0 by setting the *select* signal of the multiplexer (MUX) to a fixed value (selecting the 0 input) in test mode. As a separate example, a non-scan flip-flop that is neither scanned nor initialized is a potential X-generator (X-source). If the flip-flop has two outputs (*Q* and *QB*), one can add two multiplexers forcing both outputs to opposite values in test mode. Alternately, if the flip-flip has an asynchronous set/reset pin, an AND/OR control point can be added to permanently force the flip-flip to 0 or 1 during test. Although an AND/OR control point can be added to force the non-scan flip-flop to a constant value, it is recommended that for

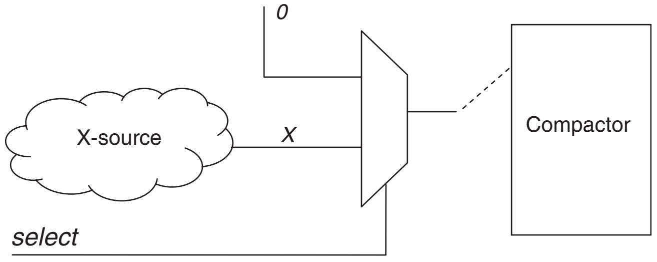


FIGURE 3.45

A simple illustration of the X-blocking scheme.

better fault coverage inserting a MUX control point driven by a nearby existing scan cell is preferred.

X-blocking can ensure that no *X*'s will be propagated to the compactor; however, it also blocks the fault effects that can only propagate to an observable point through the now-blocked X-source (*e.g.*, the non-scan flip-flop). This can result in fault coverage loss. This problem can be addressed by use of a more flexible control on the *select* signal such that the X-source is blocked only during the cycles at which it may generate *X*'s. Alternately, if the number of such faults for a given bounded X-generator justifies the cost, one or more observation points can be added before the X-source (*e.g.*, at the *D* input of the non-scan flip-flop) to provide an observable point to which those faults can propagate. These X-blocking or X-bounding methods have been extensively discussed in [Wang 2006a].

In this subsection, we only present some compactor designs that are widely used in industry along with some emerging compactors. For more information, refer to the key references cited in [Patel 2003; Mitra 2004b; Rajski 2004; Volkerink 2005; Wang 2006a; Touba 2007; Wohl 2007b].

3.5.2.1 *Combinational compaction*

A combinational compactor uses a combinational circuit to compact m outputs of the circuit under test into n test outputs, where $n < m$. If each output sequence contains only known (non-*X*) values (0's and 1's), then a combinational compactor that uses XOR gates with each internal scan chain output connected to only one XOR gate input is sufficient to guarantee no-fault coverage loss when the number of errors appearing at the m outputs is always odd [Saluja 1983]. A compactor that uses such XOR gates is referred to as a **conventional combinational compactor** or **simple space compactor**. An example is illustrated in Figure 3.46 [Wang 2008]. On the contrary, if any output sequence contains unknown values (*X*'s), the combinational compaction scheme must have the capability to mask off or tolerate unknowns to prevent faults from going undetected. A compactor able to mask off or tolerate *X*'s is referred to as an **X-tolerant combinational compactor** or **X-tolerant space compactor**. Two representative schemes currently practiced in industry are discussed in the following: (1) X-compact and (2) X-impact. Other schemes to further tolerate the amount of *X*'s can be found in [Patel 2003; Rajski 2004; Wohl 2004, 2007b; Wang 2008].

3.5.2.1.1 X-compact

X-compact [Mitra 2004a] is an **X-tolerant space compaction** technique that connects each internal scan chain output to two or more external scan output ports through a network of XOR gates to tolerate unknowns. A response compaction circuit designed by use of the X-compact technique is called an **X-compactor**. Figure 3.47 shows an X-compactor with eight inputs and five outputs. It is composed of four 3-input XOR gates and eleven 2-input XOR gates.

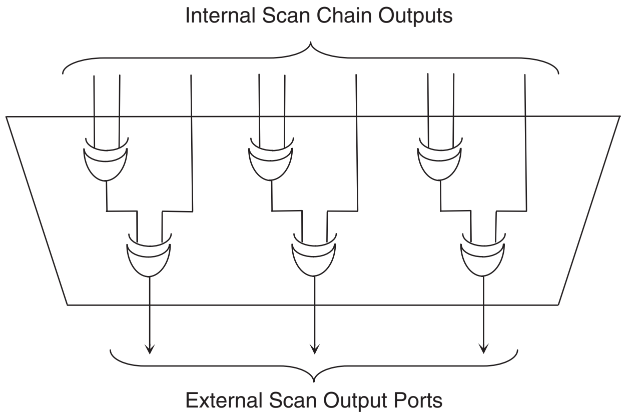


FIGURE 3.46

A conventional combinational compactor with nine inputs and three outputs.

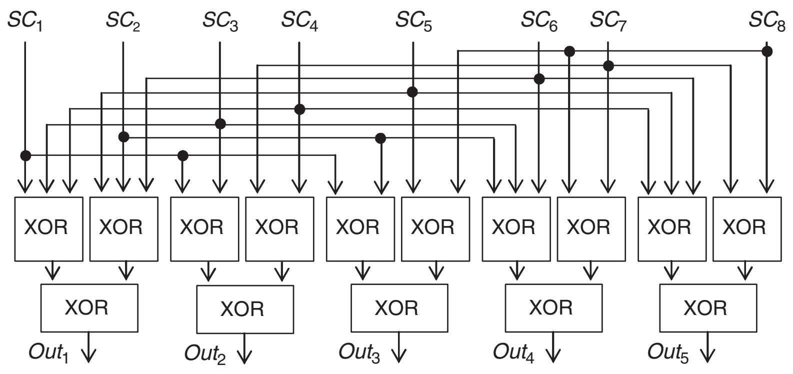


FIGURE 3.47

An X-compactor with eight inputs and five outputs.

Only one aliasing source, namely *combinational cancellation*, can exist in an X-compactor because of its combinational property. As an extreme example, if an X-compactor has only one output, it is, indeed, a **parity checker**, and any two error bits occurring simultaneously from the internal scan chain outputs will lead to aliasing.

Although aliasing may still exist when the X-compact technique is used, one can design an X-compactor that guarantees zero-aliasing in many practical cases. Consider Figure 3.47 again. If only one error bit occurs at the *SC* inputs, the error will be propagated to some output of the compactor and thus detected. One can also find that the compactor can detect any two or any odd number of errors that occur at the same cycle. In the following we use a binary matrix, called an **X-compact matrix**, to represent an X-compactor and to illustrate the fault detectability and X-tolerability of the compactor.

Suppose that the outputs of m scan chains are to be compacted into n bits for each scan cycle with an X-compactor. The associated *X-compact matrix* then contains n rows and k columns, in which each row corresponds to a scan chain output (*e.g.*, *SC* in Figure 3.47), and each column corresponds to an X-compactor output (*e.g.*, *Out* in Figure 3.47). The entry at row i and column j of the matrix is 1 if and only if the jth X-compactor output depends on the ith scan chain output; otherwise, the matrix entry is 0. Thus, the corresponding X-compact matrix M of the X-compactor shown in Figure 3.47 is:

$$M = \begin{bmatrix} 1 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 \\ 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 \end{bmatrix}$$

With the help of an X-compact matrix, it was shown in [Mitra 2004a] that errors from any one, two, or an odd number of scan chains at the same scan-out cycle are guaranteed to be detected by an X-compactor if every row of the corresponding X-compact matrix of the compactor is distinct and contains an odd number of 1's. This can be proved by the observation that (1) if all rows of the X-compact matrix are distinct, then a bitwise XOR of any two rows is nonzero, and (2) if each row further contains an odd number of 1's, then the bitwise XOR of any odd number of rows also contains an odd number of 1's.

The most distinctive feature of the X-compact technique is its X-tolerant capability (*i.e.*, detecting error bits even when the scan chain outputs have unknown bits). Refer to Figure 3.47 again. If one unknown bit occurs at SC_1, then the unknown value will be spread to Out_1, Out_2, and Out_3. Thus, after the XOR operation, the values at Out_1, Out_2, and Out_3 are masked (becoming unknown). However, if there is only one error bit in all other scan chain outputs, then the error bit will still be detected, because the error bit will be spread to at least one output that is not Out_1, Out_2, or Out_3. For example, an error bit occurring at SC_2 will be detected from Out_4. Thus, we have the following X-tolerant theorem:

Theorem 3.1:
An error from any scan chain with one unknown bit from any other scan chain at the same cycle is guaranteed to be observed at the outputs of an X-compactor if and only if:

1. No row of the X-compact matrix contains all 0's.
2. For any X-compact matrix row, the submatrix obtained by removing the row responding to the scan chain output with unknown bit and all columns having 1's in that row does not contain a row with all 0's.

The X-compact matrix of Figure 3.47 satisfies the preceding theorem. For example, if we remove row 1 and columns 1, 2, and 3, then each of the remaining rows in the submatrix contains at least a 1. Theorem 3.1 can be further extended to deal with errors from any k_1 or fewer scan chains with unknown bits from any k_2 or fewer scan chains ($k_1 + k_2 \leq n$) as follows:

Theorem 3.2:
Errors from any k_1 or fewer scan chains with unknown bits from any k_2 or fewer scan chains at the same cycle, where $k_1 + k_2 \leq n$ and n is the number of scan chains, are guaranteed to be observed at the outputs of an X-compactor if and only if:

1. No row of the X-compact matrix contains all 0's.
2. For any set S of k_1 X-compact matrix rows, any set of k_2 rows in the submatrix obtained by removing the rows in S and the X-compact matrix columns having 1's in the rows in S are linearly independent.

Designing an X-compact matrix to satisfy Theorem 3.2 is a complicated problem when an X-compactor is expected to tolerate three or more unknown bits. In some cycles, the number of actual knowns appearing at the scan chain outputs could exceed the number of unknowns designed to be tolerated by the X-compactor. Hence, the fault detectability and X-tolerability of an X-compactor highly depends on its actual implementation and the number of unknowns to be tolerated.

3.5.2.1.2 X-impact

Although X-blocking and X-compact each can achieve significant reduction in fault coverage loss caused by X's present at the inputs of a combinational compactor, the **X-impact** technique described in [Wang 2004] is helpful in that it can further reduce fault coverage loss simply by use of ATPG to algorithmically handle the impact of residual X's on the combinational compactor without adding any extra circuitry. The combinational compactor in use can be either a conventional combinational compactor or an X-tolerant combinational compactor.

Example 3.1 An example of algorithmically handling X-impact is shown in Figure 3.48. Here, SC_1 to SC_4 are scan cells connected to a conventional combinational compactor composed of XOR gates G_7 and G_8. Lines a, b, ..., h are internal signals, and line f is assumed to be connected to an X-source (memory, non-scan storage element, etc.). Now consider the detection of the stuck-at-0 (SA0) fault f_1. Logic value 1 should be assigned to both lines d and e to activate f_1. The fault effect will be captured by scan cell SC_3. If the X on f propagates to SC_4, then the compactor output q will become X and f_1 cannot be detected. To avoid this, ATPG can try to assign either 1 to line g or 0 to line h to block the X from reaching SC_4. If it is impossible to achieve this assignment, ATPG can then try to assign 1 to line c, 0 to line b, and 0 to line a to propagate the fault effect to SC_2. As a result, fault f_1 can be detected. Thus, X-impact is avoided by algorithmic assignment without adding any extra circuitry.

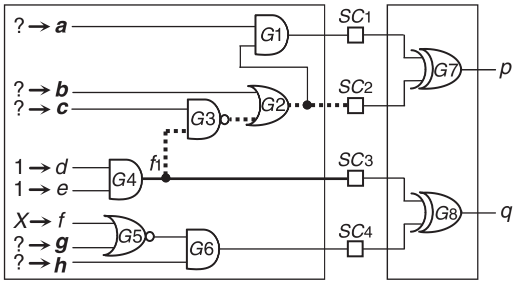


FIGURE 3.48

Handling of X-impact.

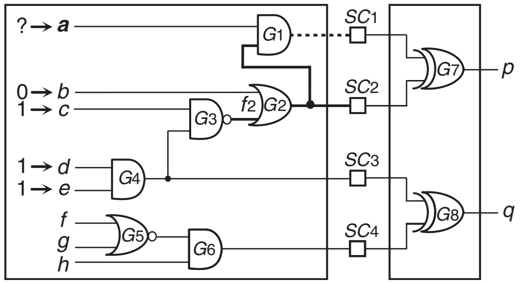


FIGURE 3.49

Handling of aliasing.

Example 3.2 It is also possible to use the X-impact approach to reduce combinational cancellation (an aliasing source). An example of algorithmically handling aliasing is shown in Figure 3.49. Here, SC_1 to SC_4 are scan cells connected to a conventional combinational compactor composed of XOR gates G_7 and G_8. Lines a, b, . . . , h are internal signals. Now consider the detection of the stuck-at-1 fault f_2. Logic value 1 should be assigned to lines c, d, and e to activate f_2, and logic value 0 should be assigned to line b to propagate the fault effect to SC_2. If line a is set to 1, then the fault effect will also propagate to SC_1. In this case, aliasing will cause the compactor output p to have a fault-free value, resulting in an undetected f_2. To avoid this, ATPG can try to assign 0 to line a to block the fault effect from reaching SC_1. As a result, fault f_2 can be detected. Thus, aliasing can be avoided by algorithmic assignment without any extra circuitry.

3.5.2.2 *Sequential compaction*

In contrast to a combinational compactor that typically uses XOR gates to compact output responses, a sequential compactor uses sequential logic instead. The sequential compactor can be a **time-space compressor** or a **space-time compressor** as described in [Saluja 1983], although the authors only considered output bit streams of 0's and 1's. The type of sequential logic to be used for response compaction depends on whether the output responses contain unknown values (*X*'s). A sequential compactor capable of masking off or tolerating these *X*'s is often referred to as an **X-tolerant sequential compactor**.

3.5.2.2.1 Signature analysis

If X-bounding as described previously has been used such that each output response does not contain any unknown (*X*) values, then the *multiple-input signature register* (MISR) widely used for logic BIST applications can be simply used [Frohwerk 1977]. Referred to as a **conventional sequential compactor**, the MISR uses an XOR gate at each MISR stage input to compact the output sequences, M_0 to M_3, into the *linear feedback shift register* (LFSR) simultaneously. The final contents stored in the MISR after compaction is often called the (*final*) *signature* of the MISR. A conventional sequential compactor that uses a four-stage MISR is illustrated in Figure 3.50. For more information on signature analysis and the MISR design, the reader is referred to Section 3.4.2.3.

3.5.2.2.2 X-masking

On the contrary, if the output response contains unknown (*X*) values, then one must make sure when the sequential compactor is used that no *X*'s from the circuit under test will reach the compactor. Although it may not result in fault coverage loss, the X-bounding scheme described previously does add area overhead and may impact delay because of the inserted logic. It is not surprising to find that, in complex designs, more than 25% of scan cycles could contain one or more *X*'s in the test response. It is difficult to eliminate these residual *X*'s by DFT; thus, an encoder with high X-tolerance is very attractive. Instead of blocking the *X*'s where they are generated, the *X*'s can also be masked off right before the sequential compactor. This scheme is referred to as **X-masking**. A typical X-masking circuit is shown in Figure 3.51. The mask controller applies a logic value 1 at the appropriate time to mask off any scan output that contains an *X* before the *X* reaches the compactor.

The **X-masking compactor** is one type of X-tolerant sequential compactors. Typically, it implies that sequential logic (comprising one or more MISRs or SISRs) is used in the compactor for response compaction. Almost all existing X-tolerant sequential compactors proposed in the literature use X-masking, including OPMISR+ [Barnhart 2002; Naruse 2003], ETCompression [Nadeau-Dostie 2004],

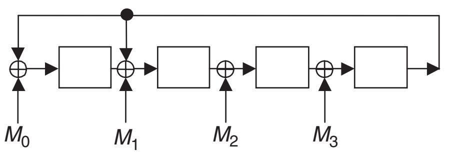


FIGURE 3.50

A conventional sequential compactor that uses a four-stage MISR.

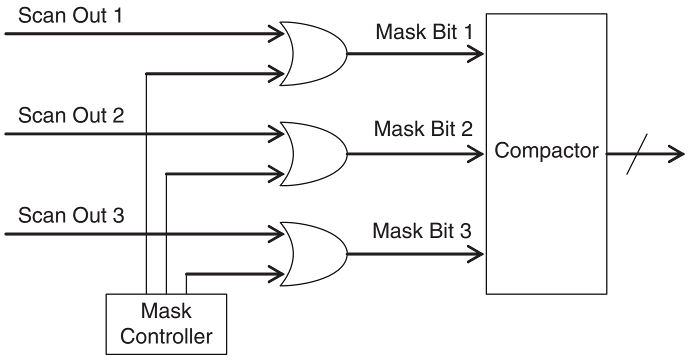


FIGURE 3.51

An example X-masking circuit in use with a compactor.

and convolutional compactors [Mitra 2004b; Rajski 2005, 2008]. In fact, combinational logic (such as XOR gates) can also be used in the compactor. Such an X-masking compactor that uses combinational logic is referred to as a **selective compactor** [Rajski 2004]. Mask data are needed to indicate when the masking should take place. These mask data can be stored in compressed format and can be decompressed with on-chip hardware. Possible compression techniques are *weighted pseudo-random LFSR reseeding* or *run-length encoding* [Volkerink 2005].

Another type of X-tolerant sequential compactor is an **X-canceling MISR** [Touba 2007, 2008] that does not mask the *X*'s before they enter the MISR. It allows the *X*'s to be compacted in a MISR and then selectively XORs together combinations of MISR signature bits that are linearly dependent in terms of the *X*'s such that all the *X*'s are canceled out.

3.5.2.2.3 *q*-compact

In case none of the X-bounding, X-masking, or X-canceling schemes is available to block, mask off, or cancel all *X*'s, the sequential logic in use must not have a feedback path so these *X*'s will only stay in the sequential compactor for a few clock cycles. Such an X-tolerant sequential compaction scheme is referred to as ***q*-compact**. A ***q*-compactor** that uses this X-tolerant compaction scheme is illustrated in [Han 2006].

Figure 3.52 shows an example of a *q*-compactor assuming the inputs are coming from internal scan chain outputs [Han 2006]. The spatial part of the *q*-compactor consists of single-output XOR networks (called *spread networks*) connected to the flip-flops by means of additional 2-input XOR gates interspersed between successive storage elements. As can be seen, every error in a scan cell can reach storage elements and then outputs in several possible ways. The spread network that determines this property is defined in terms of

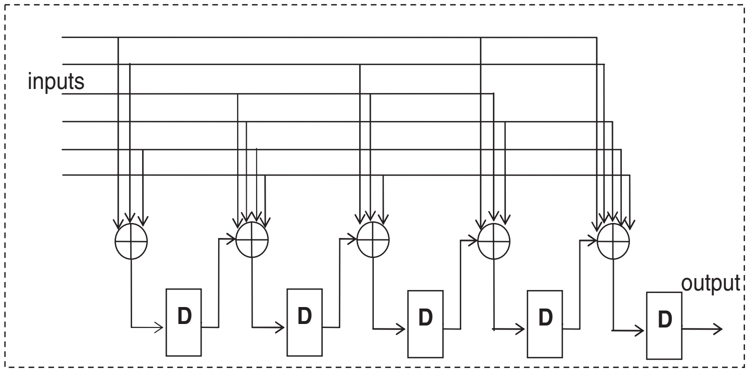


FIGURE 3.52

An example *q*-compactor with single output.

spread polynomials indicating how particular scan chains are connected to the register flip-flops.

Different from a conventional MISR, the *q*-compactor presented in Figure 3.52 does not have a feedback path; consequently, any error or *X* injected into the compactor is shifted out after at most five clock cycles. The shifted-out data will be compared with the expected data and then the error will be detected.

3.5.3 Industry practices

Several test compression products and solutions have been introduced by some of the major DFT vendors in the CAD industry. These products differ significantly with regard to technology, design overhead, design rules, and the ease of use and implementation. A few second-generation products have also been introduced by a few of the vendors [Kapur 2008]. This subsection summarizes a few of the products introduced by companies such as Cadence Design Systems [Cadence 2008], LogicVision [LogicVision 2008], Mentor Graphics [Mentor 2008], Synopsys [Synopsys 2008], and SynTest Technologies [SynTest 2008].

Current industry solutions can be grouped under two main categories for stimulus decompression. The first category uses *linear-decompression-based schemes*, whereas the second category uses *broadcast-scan-based schemes*. The main difference between the two categories is the manner in which the ATPG engine is used. The first category includes products, such as ETCompression [LogicVision 2008] from LogicVision, TestKompress [Rajski 2004] from Mentor Graphics, XOR Compression [Cadence 2008] from Cadence, and SOCBIST [Wohl 2003] from Synopsys. The second category includes products, such as OPMISR+ [Barnhart 2002; Cadence 2008] from Cadence, VirtualScan [Wang 2004, 2008] from SynTest, and DFT MAX [Sitchinava 2004; Wohl 2007a] from Synopsys.

For designs that use *linear-decompression-based schemes*, test compression is achieved in two distinct steps. During the first step, conventional ATPG is used to generate sparse ATPG patterns (called test cubes), in which *dynamic compaction* is performed in a nonaggressive manner, while leaving unspecified bit locations in each test cube as *X*. This is accomplished by not aggressively performing the *random fill* operation on the test cubes, which is used to increase coverage of individual patterns, and hence reduce the total pattern count. During the second step, a system of linear equations, describing the hardware mapping from the external scan input ports to the internal scan chain inputs, are solved to map each test cube into a compressed stimulus that can be applied externally. If a mapping is not found, a new attempt at generating a new test cube is required.

For designs that use *broadcast-scan-based schemes*, only a single step is required to perform test compression. This is achieved by embedding the constraints introduced by the decompressor as part of the ATPG tool, such that the tool operates with much more restricted constraints. Hence, whereas in conventional ATPG, each individual scan cell can be set to 0 or 1 independently, for *broadcast-scan-based schemes* the values to which related scan cells can be set are constrained. Thus, a limitation of this solution is that in some cases, the constraints among scan cells can preclude some faults from being tested. These faults are typically tested as part of a later top-up ATPG process if required, similar to the use of *linear-decompression-based schemes*.

On the response compaction side, industry solutions have used either combinational compactors such as XOR networks, or sequential compactors such as MISRs, to compact the test responses. At present, combinational compactors have a higher acceptance rate in the industry because they do not involve the process of guaranteeing that no unknown (*X*) values are generated in the circuit under test.

A summary of the different compression architectures used in the commercial products is shown in Table 3.8. Six products from five DFT companies are included. Since June 2006, Cadence has added XOR Compression as an alternative to the OPMISR+ product described in [Wang 2006a].

Table 3.8 Summary of Industry Practices for Test Compression

Industry Practices	**Stimulus Decompressor**	**Response Compactor**
XOR Compression or OPMISR+	Combinational XOR Network or Fanout Network	XOR Network with or without MISR
TestKompress	Ring Generator	XOR Network
VirtualScan	Combinational Logic Network	XOR Network
DFT MAX	Combinational MUX Network	XOR Network
ETCompression	(Reseeding) PRPG	MISR

Table 3.9 Summary of Industry Practices for At-Speed Delay Fault Testing

Industry Practices	Skewed-Load	Double-Capture
XOR Compression or OPMISR+	√	√
TestKompress	√	√
VirtualScan	√	√
DFT MAX	√	√
ETCompression	√	Through Service

It is evident that the solutions offered by the current EDA DFT vendors are quite diverse with regard to stimulus decompression and response compaction. For stimulus decompression, OPMISR+, VirtualScan, and DFT MAX are broadcast-scan–based, whereas TestKompress and ETCompression are linear-decompression–based. For response compaction, OPMISR+ and ETCompression can include MISRs, whereas four other solutions purely adopt (X-tolerant) XOR networks. What is common is that all six products provide their own diagnostic solutions.

Generally speaking, any modern ATPG compression program supports at-speed clocking schemes used in its corresponding at-speed scan architecture. For at-speed delay fault testing, ETCompression currently uses a **skewed-load–based at-speed test compression architecture** for ATPG. The product can also support the double-capture clocking scheme through service. All other ATPG compression products, including OPMISR+, TestKompress, VirtualScan, and DFT MAX, support the **hybrid at-speed test compression architecture** by use of both skewed-load (*a.k.a.* launch-on-shift) and double-capture (*a.k.a.* launch-on-capture). In addition, almost every product supports inter-clock-domain delay fault testing for synchronous clock domains. A few on-chip clock controllers for detecting these inter-clock-domain delay faults at-speed have been proposed in [Beck 2005; Nadeau-Dostie 2005, 2006; Furukawa 2006; Fan 2007; and Keller 2007].

The clocking schemes used in these commercial products are summarized in Table 3.9. It should be noted that compression schemes might be limited in effectiveness if there are a large number of unknown response values, which can be exacerbated during at-speed testing when many paths do not make the timing being used.

3.6 CONCLUDING REMARKS

Design for testability (DFT) has become vital for ensuring circuit testability and product quality. Scan design, which has proven to be the most powerful DFT technique ever invented, allowed the transformation of sequential circuit testing into combinational circuit testing and has since become an industry standard. Currently, a scan design can contain a billion transistors [Naffziger 2006; Stackhouse 2008]. To screen all possible physical failures (manufacturing defects) caused by manufacturing imperfection, test compression coupled to scan design has rapidly emerged, becoming a crucial DFT technique to address the explosive test data volume and long test application time problems. At the same time, scan-based logic *built-in self-test* (BIST) is of growing importance because of its inherent advantage of performing self-test on-chip, on-board, or in-system, which can substantially improve the reliability of the system and the ability of in-field diagnosis.

Whereas the STUMPS-based architecture [Bardell 1982] is the most popular logic BIST architecture practiced currently for scan-based designs, the efforts required to implement the BIST circuitry and the loss of the fault coverage for the use of pseudo-random patterns have prevented the BIST architecture from being widely used in industry. As the semiconductor manufacturing technology moves into the nanometer design era, it remains to be seen how the CBILBO-based architecture proposed in [Wang 1986b], which can always guarantee 100% single stuck-at fault coverage and has the ability of running 10 times more BIST patterns than the STUMPS-based architecture, will perform. Challenges lie ahead with regard to whether or not pseudo-exhaustive testing will become a preferred BIST pattern generation technique.

Because the primary objective of this chapter is to familiarize the reader with basic DFT techniques, many advanced DFT techniques, along with novel *design-for-reliability* (DFR), *design-for-manufacturability* (DFM), *design-for-yield* (DFY), *design-for-debug-and-diagnosis* (DFD), and low-power test techniques, are left out. For advanced reading, the reader is referred to [Gizopoulos 2006; Wang 2006a, 2007a]. These techniques are of growing importance to help us cope with the physical failures of the nanometer design era.

The DFT chapter is the first of a series of three chapters devoted to VLSI testing. These chapters are chosen to equip the reader with basic DFT skills to design quality digital circuits. Chapter 7 discusses the design rules and test synthesis steps required to implement testability logic into these digital circuits. Chapter 14 jumps into the important fault simulation and test generation techniques for generating quality test patterns to screen defective chips from manufacturing test.

3.7 EXERCISES

3.1. **(Testability Analysis)** Calculate the SCOAP controllability and observability measures for a 3-input XOR gate and for its NAND-NOR implementation.

3.2. **(Testability Analysis)** Use the rules given in Tables 3.3 and 3.4 to calculate the probability-based testability measures for a 3-input XNOR gate and for its NAND-NOR implementation. Assume that the probability-based controllability values at all primary inputs and the probability-based observability value at the primary output are 0.5 and 1, respectively.

3.3. (**Testability Analysis**) Repeat Exercise 3.2 for the full-adder circuit shown in Figure 3.1.

3.4. (**Muxed-D Scan Cell**) Show a possible CMOS implementation of the muxed-D scan cell shown in Figure 3.5a.

3.5. (**Low-Power Muxed-D Scan Cell**) Design a low-power version of the muxed-D scan cell given in Figure 3.5a by adding gated-clock logic that includes a lock-up latch to control the clock port.

3.6. (**At-Speed Scan**) Assume that a scan design contains three clock domains running at 100 MHz, 200 MHz, and 400 MHz, respectively. In addition, assume that the clock skew between any two clock domains is manageable. List all possible at-speed scan ATPG methods and compare their advantages and disadvantages in terms of fault coverage and test pattern count.

3.7. (**At-Speed Scan**) Describe two major capture-clocking schemes for at-speed scan testing and compare their advantages and disadvantages. Also discuss what will happen if three or more captures are used.

3.8. (**BIST Pattern Generation**) Implement a period-8 in-circuit *test pattern generator* (TPG) with a binary counter. Compare its advantages and disadvantages with a Johnson counter (twisted-ring counter).

3.9. (**BIST Pattern Generation**) Implement a period-31 in-circuit *test pattern generator* (TPG) with a modular *linear feedback shift register* (LFSR) with *characteristic polynomial* $f(x) = 1 + x^2 + x^5$. Convert the modular LFSR into a muxed-D scan design with minimum area overhead.

3.10. (**BIST Pattern Generation**) Implement a period-31 in-circuit *test pattern generator* (TPG) with a five-stage *cellular automaton* (CA) with *construction rule = 11001, where* "0" denotes a *rule 90* cell and "1" denotes a *rule 150* cell. Convert the CA into an LSSD design with minimum area overhead.

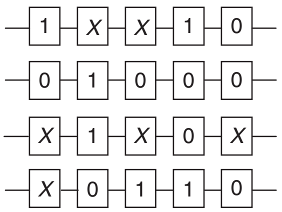
3.11. (**Cellular Automata**) Derive a construction rule for a cellular automaton of length 54, and then construction rules up to length 300 to match the list of primitive polynomials up to degree 300 reported in [Bardell 1987].

3.12. (**BIST Response Compaction**) Discuss in detail what errors can and cannot be detected by a MISR.

3.13. (**STUMPS *versus* CBILBO**) Compare the performance of a STUMPS design and a CBILBO design. Assume that both designs operate at 400 MHz and that the circuit under test has 100 scan chains each having 1000 scan cells. Compute the test time for each design when 100,000 test patterns are to be applied. In general, the shift (scan) speed is much slower than a circuit's operating speed. Assume that the scan shift frequency is 50 MHz, and compute the test time for the STUMPS design again. Explain further why the STUMPS-based architecture is gaining more popularity than the CBILBO-based architecture.

3.14. (**Scan *versus* Logic BIST *versus* Test Compression**) Compare the advantages and disadvantages of a scan design, a logic BIST design, and a test compression design in terms of fault coverage, test application time, test data volume, and area overhead.

3.15. (**Test Stimulus Compression**) Given a circuit with four scan chains, each having five scan cells, and with a set of test cubes listed:



a. Design the multiple-input broadcast scan decompressor that fulfills the test cube requirements.

b. What is the compression ratio?

c. The assignment of X's will affect the compression performance dramatically. Give one X-assignment example that will unfortunately lead to no compression with this multiple-input broadcast scan decompressor.

3.16. (**Test Stimulus Compression**) Derive mathematical expressions for the following in terms of the number of tester channels, n, and the expansion ratio, k.

a. The probability of encoding a scan slice containing 2 specified bits with Illinois scan.

b. The probability of encoding a scan slice containing 3 specified bits, where each scan chain is driven by the XOR of a unique combination of 2 tester channels such that there are a total of $C_2^n = n(n-1)/2$ scan chains.

3.17. (**Test Stimulus Compression**) For the sequential linear decompressor shown in Figure 3.38 whose corresponding system of linear equations is shown in Figure 3.39, find the compressed stimulus, $X_1 - X_{10}$, necessary to encode the following test cube: $< Z_1, \ldots, Z_{12} >$ = <0---1-0--010>.

3.18. (**Test Stimulus Compression**) For the MUX network shown in Figure 3.43 and then the XOR network shown in Figure 3.44a, find the compressed stimulus at the network inputs necessary to encode the following test cube: <1 - 0 - - - 01>.

3.19. **(Test Response Compaction)** Explain further how many errors and how many unknowns (X's) can be detected or tolerated by the X-compactor and q-compactor as shown in Figures 3.47 and 3.52, respectively.

3.20. **(Test Response Compaction)** For the X-compact matrix of the X-compactor given below:

$$\begin{bmatrix} 0 & 1 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 & 1 \\ 1 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 & 1 \end{bmatrix}$$

a. What is the compaction ratio?

b. Which outputs after compaction are affected by the second scan chain output?

c. How many errors can be detected by the X-compactor?

ACKNOWLEDGMENTS

I wish to thank Dr. Xinghao Chen of CTC Technologies for contributing the Testability Analysis section; Professor Xiaowei Li and Professor Yinhe Han of Chinese Academy of Sciences, Professor Kuen-Jong Lee of National Cheng Kung University, Professor Nur A. Touba of the University of Texas at Austin for contributing a portion of the Circuits for Test Stimulus Compression and Circuits for Test Response Compaction sections. I also express my gratitude to Professor Xiaoqing Wen of Kyushu Institute of Technology, Professor Nur A. Touba of the University of Texas at Austin, Professor Kewal K. Saluja of the University of Wisconsin–Madison, Professor Subhasish Mitra of Stanford University, Dr. Rohit Kapur and Khader S. Abdel-Hafez of Synopsys, Dr. Brion Keller of Cadence Design Systems, and Dr. Benoit Nadeau-Dostie of LogicVision for reviewing the text and providing helpful comments, and Teresa Chang of SynTest Technologies for drawing most of the figures.

REFERENCES

R3.0 Books

[Abramovici 1994] M. Abramovici, M. A. Breuer, and A. D. Friedman, *Digital Systems Testing and Testable Design,* IEEE Press, Revised Printing, Piscataway, NJ, 1994.

[Bardell 1987] P. H. Bardell, W. H. McAnney, and J. Savir, *Built-In Test for VLSI: Pseudorandom Techniques,* John Wiley & Sons, Somerset, NJ, 1987.

[Bushnell 2000] M. L. Bushnell and V. D. Agrawal, *Essentials of Electronic Testing for Digital, Memory & Mixed-Signal VLSI Circuits,* Springer, Boston, 2000.

[Crouch 1999] A. Crouch, *Design for Test for Digital IC's and Embedded Core Systems,* Prentice-Hall, Englewood Cliffs, NJ, 1999.

[Gizopoulos 2006] D. Gizopoulos, editor, *Advances in Electronic Testing: Challenges and Methodologies,* Morgan Kaufmann, San Francisco, 2006.

[Golomb 1982] S. W. Golomb, *Shift Register Sequence,* Aegean Park Press, Laguna Hills, CA, 1982.

[Jha 2003] N. Jha and S. Gupta, *Testing of Digital Systems,* Cambridge University Press, London, 2003.

[McCluskey 1986] E. J. McCluskey, *Logic Design Principles: With Emphasis on Testable Semicustom Circuits,* Prentice-Hall, Englewood Cliffs, NJ, 1986.

[Mourad 2000] S. Mourad and Y. Zorian, *Principles of Testing Electronic Systems,* John Wiley & Sons, Somerset, NJ, 2000.

[Nadeau-Dostie 2000] B. Nadeau-Dostie, *Design for At-Speed Test, Diagnosis and Measurement,* Springer, Boston, 2000.

[Peterson 1972] W. W. Peterson and E. J. Weldon, Jr., *Error-Correcting Codes,* MIT Press, Cambridge, MA, 1972.

[Rajski 1998] J. Rajski and J. Tyszer, *Arithmetic Built-In Self-Test for Embedded Systems,* Prentice-Hall, Englewood Cliffs, NJ, 1998.

[Stroud 2002] C. E. Stroud, *A Designer's Guide to Built-In Self-Test,* Springer, Boston, 2002.

[Wang 2006a] L.-T. Wang, C.-W. Wu, and X. Wen, editors, *VLSI Test Principles and Architectures: Design for Testability,* Morgan Kaufmann, San Francisco, 2006.

[Wang 2007a] L.-T. Wang, C. E. Stroud, and N. A. Touba, editors, *System-on-Chip Test Architectures: Nanometer Design for Testability,* Morgan Kaufmann, San Francisco, 2007.

R3.1 Introduction

[Fujiwara 1982] H. Fujiwara and S. Toida, The complexity of fault detection problems for combinational circuits, *IEEE Trans. on Computers*, C-31(6), pp. 555–560, June 1982.

[SIA 2005] SIA, *The International Technology Roadmap for Semiconductors: 2005 Edition—Design,* Semiconductor Industry Association, San Jose, CA, http://public.itrs.net, 2005.

[SIA 2006] SIA, *The International Technology Roadmap for Semiconductors: 2006 Update,* Semiconductor Industry Association, San Jose, CA, http://public.itrs.net, 2006.

[Touba 2006] N. A. Touba, Survey of test vector compression techniques, *IEEE Design & Test of Computers*, 23(4), pp. 294–303, July–August 2006.

R3.2 Testability Analysis

[Agrawal 1982] V. D. Agrawal and M. R. Mercer, Testability measures—What do they tell us?, in *Proc. IEEE Int. Test Conf.*, pp. 391–396, November 1982.

[Breuer 1978] M. A. Breuer, New concepts in automated testing of digital circuits, in *Proc. EEC Symp. on CAD of Digital Electronic Circuits and Systems*, pp. 69–92, November 1978.

[Goldstein 1979] L. H. Goldstein, Controllability/Observability analysis of digital circuits, *IEEE Trans. on Circuits and Systems*, CAS-26(9), pp. 685–693, September 1979.

[Goldstein 1980] L. H. Goldstein and E. L. Thigpen, SCOAP: Sandia controllability/observability analysis program, in *Proc. ACM/IEEE Design Automation Conf.*, pp. 190–196, June 1980.

[Grason 1979] J. Grason, TMEAS—a testability measurement program, in *Proc. ACM/IEEE Design Automation Conf.*, pp. 156–161, June 1979.

[Ivanov 1988] A. Ivanov and V. K. Agarwal, Dynamic testability measures for ATPG, *IEEE Trans. on Computer-Aided Design*, 7(5), pp. 598–608, May 1988.

[Jain 1985] S. K. Jain and V. D. Agrawal, Statistical fault analysis, *IEEE Design & Test of Computers*, 2(2), pp. 38–44, February 1985.

[Parker 1975] K. P. Parker and E. J. McCluskey, Probability treatment of general combinational networks, *IEEE Trans. on Computers*, 24(6), pp. 668–670, June 1975.

[Rizzolo 2001] R. F. Rizzolo, B. F. Robbins, and D. G. Scott, A hierarchical approach to improving random pattern testability on IBM eServer z900 chips, in *Digest of Papers, IEEE North Atlantic Test Workshop*, pp. 84–89, May 2001.

[Rutman 1972] R. A. Rutman, Fault detection test generation for sequential logic heuristic tree search, *IEEE Computer Repository,* Paper R-72-187, September/October 1972.

[Savir 1984] J. Savir, G. S. Ditlow, and P. H. Bardell, random pattern testability, *IEEE Trans. on Computer*, C-33(1), pp. 79–90, January 1984.

[Seth 1985] S. C. Seth, L. Pan, and V. D. Agrawal, PREDICT—Probabilistic estimation of digital circuit testability, in *Proc. IEEE Fault-Tolerant Computing Symp.*, pp. 220–225, June 1985.

[Stephenson 1976] J. E. Stephenson and J. Garson, A testability measure for register transfer level digital circuits, in *Proc. IEEE Fault-Tolerant Computing Symp.*, pp. 101–107, June 1976.

[Wang 1984] L.-T. Wang and E. Law, Daisy testability analyzer (DTA), in *Proc. IEEE/ACM Int. Conf. on Computer-Aided Design*, pp. 143–145, November 1984.

[Wang 1985] L.-T. Wang and E. Law, An enhanced Daisy testability analyzer (DTA), in *Proc. Automatic Testing Conf.*, pp. 223–229, October 1985.

R3.3 Scan Design

[Cheung 1997] B. Cheung and L.-T. Wang, The seven deadly sins of scan-based designs, in *Integrated System Design*, www.eetimes.com/editorial/1997/test9708.html, August 1997.

[DasGupta 1982] S. DasGupta, P. Goel, R. G. Walther, and T. W. Williams, A variation of LSSD and its implications on design and test pattern generation in VLSI, in *Proc. IEEE Int. Test Conf.*, pp. 63–66, November 1982.

[Eichelberger 1977] E. B. Eichelberger and T. W. Williams, A logic design structure for LSI testability, in *Proc. ACM/IEEE Design Automation Conf.*, pp. 462–468, June 1977.

[Nadeau-Dostie 1994] B. Nadeau-Dostie, A. Hassan, D. Burek, and S. Sunter, Multiple Clock Rate Test Apparatus for Testing Digital Systems, U.S. Patent No. 5,349,587, September 20, 1994.

[Rajski 2003] J. Rajski, A. Hassan, R. Thompson, and N. Tamarapalli, Method and Apparatus for At-Speed Testing of Digital Circuits, U.S. Patent Application No. 20030097614, May 22, 2003.

[Savir 1993] J. Savir and S. Patil, Scan-based transition test, *IEEE Trans. on Computer-Aided Design*, 12(8), pp. 1232–1241, August 1993.

[Savir 1994] J. Savir and S. Patil, Broad-side delay test, *IEEE Trans. on Computer-Aided Design*, 13(8), pp. 1057–1064, August 1994.

[Wang 2005a] L.-T. Wang, M.-C. Lin, X. Wen, H.-P. Wang, C.-C. Hsu, S.-C. Kao, and F.-S. Hsu, Multiple-Capture DFT System for Scan-Based Integrated Circuits, U.S. Patent No. 6,954,887, October 11, 2005.

[Wang 2007b] L.-T. Wang, P.-C. Hsu, and X. Wen, Multiple-Capture DFT System for Detecting or Locating Crossing Clock-Domain Faults During Scan-Test, U.S. Patent No. 7,260,756, August 21, 2007.

R3.4 Logic Built-In Self-Test

[Bardell 1982] P. H. Bardell and W. H. McAnney, Self-testing of multiple logic modules, in *Proc. IEEE Int. Test Conf.*, pp. 200–204, November 1982.

[Barzilai 1981] Z. Barzilai, J. Savir, G. Markowsky, and M. G. Smith, The weighted syndrome sums approach to VLSI testing, *IEEE Trans. on Computers*, 30(12), pp. 996–1000, December 1981.

[Barzilai 1983] Z. Barzilai, D. Coppersmith, and A. Rosenberg, Exhaustive bit pattern generation in discontiguous positions with applications to VLSI testing, *IEEE Trans. on Computers*, 32(2), pp. 190–194, February 1983.

[Benowitz 1975] N. Benowitz, D. F. Calhoun, G. E. Alderson, J. E. Bauer, and C. T. Joeckel, An advanced fault isolation system for digital logic, *IEEE Trans. on Computers*, 24(5), pp. 489–497, May 1975.

[Cadence 2008] Cadence Design Systems, http://www.cadence.com, 2008.

[Chen 1987] C. L. Chen, Exhaustive test pattern generation with cyclic codes, *IEEE Trans. on Computers*, 37(3), pp. 329–338, March 1987.

[Cheon 2005] B. Cheon, E. Lee, L.-T. Wang, X. Wen, P. Hsu, J. Cho, J. Park, H. Chao, and S. Wu, At-speed logic BIST for IP cores, in *Proc. IEEE/ACM Design, Automation, and Test in Europe Conf.*, pp. 860–861, March 2005.

[Chin 1984] C. K. Chin and E. J. McCluskey, *Weighted Pattern Generation for Built-In Self-Test*, Center for Reliable Computing, Technical Report (CRC TR) No. 84-7, Stanford University, August 1984.

[Foote 1997] T. G. Foote, D. E. Hoffman, W. V. Huott, T. J. Koprowski, B. J. Robbins, and M. P. Kusko, Testing the 400 MHz IBM generation-4 CMOS chip, in *Proc. IEEE Int. Test Conf.*, pp. 106–114, November 1997.

[Frohwerk 1977] R. A. Frohwerk, Signature analysis: A new digital field service method, in *Hewlett-Packard J.*, 28, pp. 2–8, September 1977.

[Furukawa 2006] H. Furukawa, X. Wen, L.-T. Wang, B. Sheu, Z. Jiang, and S. Wu, A novel and practical control scheme for inter-clock at-speed testing, in *Proc. IEEE Int. Test Conf.*, Paper 17.2, October 2006.

[Gloster 1988] C. S. Gloster, Jr. and F. Brglez, Boundary scan with cellular built-in self-test, in *Proc. IEEE Int. Test Conf.*, pp. 138–145, September 1988.

[Hassan 1984] S. Z. Hassan and E. J. McCluskey, Increased fault coverage through multiple signatures, in *Proc. IEEE Fault-Tolerant Computing Symp.*, pp. 354–359, June 1984.

[Hayes 1976] J. P. Hayes, Transition count testing of combinational logic circuits, *IEEE Trans. on Computers*, C-25(6), pp. 613–620, June 1976.

[Hortensius 1989] P. D. Hortensius, R. D. McLeod, W. Pries, D. M. Miller, and H. C. Card, Cellular automata-based pseudorandom number generators for built-in self-test, *IEEE Trans. on Computer-Aided Design*, 8(8), pp. 842–859, August 1989.

[Keller 2007] B. Keller, A. Uzzaman, B. Li, and T. Snethen, Using programmable on-product clock generation (OPCG) for delay test, in *Proc. IEEE Asian Test Symp.*, pp. 69–72, October 2007.

[Khara 1987] M. Khara and A. Albicki, Cellular automata used for test pattern generation, in *Proc. IEEE Int. Conf. on Computer Design*, pp. 56–59, October 1987.

[Könemann 1979] B. Könemann, J. Mucha, and G. Zwiehoff, Built-in logic block observation techniques, in *Proc. IEEE Int. Test Conf.*, pp. 37–41, October 1979.

[Könemann 1980] B. Könemann, J. Mucha, and G. Zwiehoff, Built-in test for complex digital circuits, *IEEE J. of Solid-State Circuits*, 15(3), pp. 315–318, June 1980.

[Lai 2007] L. Lai, W.-T. Cheng, and T. Rinderknecht, Programmable scan-based logic built-in self test, in *Proc. IEEE Asian Test Symp.*, pp. 371–377, October 2007.

[LogicVision 2008] LogicVision, http://www.logicvision.com, 2008.

[McCluskey 1981] E. J. McCluskey and S. Bozorgui-Nesbat, Design for autonomous test, *IEEE Trans. on Computers*, 30(11), pp. 860–875, November 1981.

[McCluskey 1984] E. J. McCluskey, Verification testing—A pseudoexhaustive test technique, *IEEE Trans. on Computers*, 33(6), pp. 541–546, June 1984.

[McCluskey 1985] E. J. McCluskey, Built-in self-test structures, *IEEE Design & Test of Computers*, 2(2), pp. 29–36, April 1985.

[Mentor 2008] Mentor Graphics, http://www.mentor.com, 2008.

[Nadeau-Dostie 1994] B. Nadeau-Dostie, A. Hassan, D. Burek, and S. Sunter, Multiple Clock Rate Test Apparatus for Testing Digital Systems, U.S. Patent No. 5,349,587, September 20, 1994.

[Nadeau-Dostie 2006] B. Nadeau-Dostie and J.-F. Côté, Clock Controller for At-Speed Testing of Scan Circuits, U.S. Patent No. 7,155,651, December 26, 2006.

[Nadeau-Dostie 2007] B. Nadeau-Dostie, Method and Circuit for At-Speed Testing of Scan Circuits, U.S. Patent No. 7,194,669, March 20, 2007.

[Rajski 2003] J. Rajski, A. Hassan, R. Thompson, and N. Tamarapalli, Method and Apparatus for At-Speed Testing of Digital Circuits, U.S. Patent Application No. 20030097614, May 22, 2003.

[Savir 1980] J. Savir, Syndrome-testable design of combinational circuits, *IEEE Trans. on Computers*, 29(6), pp. 442–451, June 1980.

[Savir 1985] J. Savir and W. H. McAnney, On the masking probability with ones count and transition count, in *Proc. IEEE/ACM Int. Conf. on Computer-Aided Design*, pp. 111–113, November 1985.

[Schnurmann 1975] H. D. Schnurmann, E. Lindbloom, and R. G. Carpenter, The weighted random test-pattern generator, *IEEE Trans. on Computers*, 24(7), pp. 695–700, July 1975.

[SynTest 2008] SynTest Technologies, http://www.syntest.com, 2008.

[Tang 1984] D. T. Tang and C. L. Chen, Logic test pattern generation using linear codes, *IEEE Trans. on Computers*, 33(9), pp. 845–850, September 1984.

[Tsai 1999] H.-C. Tsai, K.-T. Cheng, and S. Bhawmik, Improving the test quality for scan-based BIST using a general test application scheme, in *Proc. ACM/IEEE Design Automation Conf.*, pp. 748–753, June 1999.

[van Sas 1990] J. van Sas, F. Catthoor, and H. D. Man, Cellular automata-based self-test for programmable data paths, in *Proc. IEEE Int. Test Conf.*, pp. 769–778, September 1990.

[Wang 1986a] L.-T. Wang and E. J. McCluskey, Condensed linear feedback shift register (LFSR) testing—A pseudoexhaustive test technique, *IEEE Trans. on Computers*, 35(4), pp. 367–370, April 1986.

[Wang 1986b] L.-T. Wang and E. J. McCluskey, Concurrent built-in logic block observer (CBILBO), in *Proc. IEEE Int. Symp. on Circuits and Systems*, 3(3), pp. 1054–1057, May 1986.

[Wang 1987] L.-T. Wang and E. J. McCluskey, Linear feedback shift register design using cyclic codes, *IEEE Trans. on Computers*, 37(10), pp. 1302–1306, October 1987.

[Wang 1988a] L.-T. Wang and E. J. McCluskey, Hybrid designs generating maximum-length sequences, *Special Issue on Testable and Maintainable Design, IEEE Trans. on Computer-Aided Design*, 7(1), pp. 91–99, January 1988.

[Wang 1988b] L.-T. Wang and E. J. McCluskey, Circuits for pseudo-exhaustive test pattern generation, *IEEE Trans. on Computer-Aided Design*, 7(10), pp. 1068–1080, October 1988.

[Wang 1989] L.-T. Wang, M. Marhoefer, and E. J. McCluskey, A self-test and self-diagnosis architecture for boards using boundary scan, in *Proc. IEEE European Test Conf.*, pp. 119–126, April 1989.

[Wang 2005b] L.-T. Wang, X. Wen, P.-C. Hsu, S. Wu, and J. Guo, At-speed logic BIST architecture for multi-clock designs, in *Proc. Int. Conf. on Computer Design*, pp. 475–478, October 2005.

[Wang 2006b] L.-T. Wang, P.-C. Hsu, S.-C. Kao, M.-C. Lin, H.-P. Wang, H.-J. Chao, and X. Wen, Multiple-Capture DFT System for Detecting or Locating Crossing Clock-Domain Faults During Self-Test or Scan-Test, U.S. Patent No. 7,007,213, February 28, 2006.

[Williams 1987] T. W. Williams, W. Daehn, M. Gruetzner, and C. W. Starke, Aliasing errors in signature analysis registers, *IEEE Design & Test of Computers*, 4(2), pp. 39–45, April 1987.

[Wolfram 1983] S. Wolfram, Statistical mechanics of cellular automata, in *Review of Modern Physics*, 55(3), pp. 601–644, July 1983.

[Wunderlich 1987] H.-J. Wunderlich, Self test using unequiprobable random patterns, in *Proc. IEEE Fault-Tolerant Computing Symp.*, pp. 258–263, July 1987.

R3.5 Test Compression

[Barnhart 2002] C. Barnhart, V. Brunkhorst, F. Distler, O. Farnsworth, A. Ferko, B. Keller, D. Scott, B. Koenemann, and T. Onodera, Extending OPMISR beyond 10x scan test efficiency, *IEEE Design & Test of Computers*, 19(5), pp. 65–73, May-June 2002.

[Bayraktaroglu 2001] I. Bayraktaroglu and A. Orailoglu, Test volume and application time reduction through scan chain concealment, in *Proc. ACM/IEEE Design Automation Conf.*, pp. 151–155, June 2001.

[Bayraktaroglu 2003] I. Bayraktaroglu and A. Orailoglu, Concurrent application of compaction and compression for test time and data volume reduction in scan designs, *IEEE Trans. on Computers*, 52(11), pp. 1480–1489, November 2003.

[Beck 2005] M. Beck, O. Barondeau, M. Kaibel, F. Poehl, X. Lin, and R. Press, Logic design for on-chip test clock generation—Implementation details and impact on delay test quality, in *Proc. IEEE/ACM Design, Automation, and Test in Europe Conf.*, pp. 56–61, March 2005.

[Cadence 2008] Cadence Design Systems, http://www.cadence.com, 2008.

[Chakrabarty 1998] K. Chakrabarty, B. T. Murray, and J. P. Hayes, Optimal zero-aliasing space compaction of test responses, *IEEE Trans. on Computers*, 47(11), pp. 1171–1187, November 1998.

[Chandra 2007] A. Chandra, H. Yan, and R. Kapur, Multimode Illinois scan architecture for test application time and test data volume reduction, in *Proc. IEEE VLSI Test Symp.*, pp. 84–92, May 2007.

[Fan 2007] X.-X. Fan, Y. Hu, and L.-T. Wang, An on-chip test clock control scheme for multi-clock at-speed testing, in *Proc. IEEE Asian Test Symp.*, pp. 341–348, October 2007.

[Frohwerk 1977] R. A. Frohwerk, Signature analysis: A new digital field service method, in *Hewlett-Packard J.*, 28, pp. 2–8, September 1977.

[Furukawa 2006] H. Furukawa, X. Wen, L.-T. Wang, B. Sheu, Z. Jiang, and S. Wu, A novel and practical control scheme for inter-clock at-speed testing, in *Proc. IEEE Int. Test Conf.*, Paper 17.2, October 2006.

[Gu 2001] X. Gu, S. S. Chung, F. Tsang, J. A. Tofte, and H. Rahmanian, An effort-minimized logic BIST implementation method, in *Proc. IEEE Int. Test Conf.*, pp. 1002–1010, October 2001.

[Hamzaoglu 1999] I. Hamzaoglu and J. H. Patel, Reducing test application time for full scan embedded cores, in *Proc. IEEE Fault-Tolerant Computing Symp.*, pp. 260–267, July 1999.

[Han 2006] Y. Han, X. Li, H. Li, and A. Chandra, Embedded test resource for SoC to reduce required tester channels based on advanced convolutional codes, *IEEE Trans. on Instrumentation and Measurement*, 55(2), pp. 389–399, April 2006.

[Han 2007] Y. Han, Y. Hu, X. Li, H. Li, and A. Chandra, Embedded test decompressor to reduce the required channels and vector memory of tester for complex processor circuit, *IEEE Trans. on Very Large Scale Integration Systems*, 15(5), pp. 531–540, May 2007.

[Hsu 2001] F. F. Hsu, K. M. Butler, and J. H. Patel, A case study on the implementation of Illinois scan architecture, in *Proc. IEEE Int. Test Conf.*, pp. 538–547, October 2001.

[Kapur 2008] R. Kapur, S. Mitra, and T. W. Williams, Historical perspective on scan compression, *IEEE Design & Test of Computers*, 25(2), pp. 114–120, March-April 2008.

[Keller 2007] B. Keller, A. Uzzaman, B. Li, and T. Snethen, Using programmable on-product clock generation (OPCG) for delay test, in *Proc. IEEE Asian Test Symp.*, pp. 69–72, October 2007.

[Könemann 1991] B. Koenemann, LFSR-coded test patterns for scan designs, in *Proc. IEEE European Test Conf.*, pp. 237–242, April 1991.

[Könemann 2001] B. Koenemann, C. Barnhart, B. Keller, T. Snethen, O. Farnsworth, and D. Wheater, A SmartBIST variant with guaranteed encoding, in *Proc. IEEE Asian Test Symp.*, pp. 325–330, November 2001.

[Könemann 2003] B. Koenemann, C. Barnhart, and B. Keller, Real-Time Decoder for Scan Test Patterns, U.S. Patent No. 6,611,933, August 26, 2003.

[Krishna 2001] C. V. Krishna, A. Jas, and N. A. Touba, Test vector encoding using partial LFSR reseeding, in *Proc. IEEE Int. Test Conf.*, pp. 885–893, October 2001.

[Krishna 2003] C. V. Krishna and N. A. Touba, Adjustable width linear combinational scan vector decompression, in *Proc. IEEE/ACM Int. Conf. on Computer-Aided Design*, pp. 863–866, September 2003.

[Lee 1998] K.-J. Lee, J. J. Chen, and C. H. Huang, Using a single input to support multiple scan chains, in *Proc. IEEE/ACM Int. Conf. on Computer-Aided Design*, pp. 74–78, November 1998.

[Lee 1999] K.-J. Lee, J. J. Chen, and C. H. Huang, Broadcasting test patterns to multiple circuits, *IEEE Trans. on Computer-Aided Design*, 18(12), pp. 1793–1802, December 1999.

[Li 2004] L. Li and K. Chakrabarty, Test set embedding for deterministic BIST using a reconfigurable interconnection network, *IEEE Trans. on Computer-Aided Design*, 23(9), pp. 1289–1305, September 2004.

[LogicVision 2008] LogicVision, http://www.logicvision.com, 2008.

[Mentor 2008] Mentor Graphics, http://www.mentor.com, 2008.

[Mitra 2004a] S. Mitra and K. S. Kim, X-Compact: An efficient response compaction technique, *IEEE Trans. on Computer-Aided Design*, 23(3), pp. 421–432, March 2004.

[Mitra 2004b] S. Mitra, S. S. Lumetta, and M. Mitzenmacher, X-tolerant signature analysis, in *Proc. IEEE Int. Test Conf.*, pp. 432–441, October 2004.

[Mitra 2006] S. Mitra and K. S. Kim, XPAND: An efficient test stimulus compression technique, *IEEE Trans. on Computers*, 55(2), pp. 163–173, February 2006.

[Mrugalski 2004] G. Mrugalski, J. Rajski, and J. Tyszer, Ring generators—new devices for embedded test applications, *IEEE Trans. on Computer-Aided Design*, 23(9), pp. 1306–1320, September 2004.

[Nadeau-Dostie 2004] B. Nadeau-Dostie, Method of Masking Corrupt Bits During Signature Analysis and Circuit for Use Therewith, U.S. Patent No. 6,745,359, June 1, 2004.

[Nadeau-Dostie 2005] B. Nadeau-Dostie, J.-F. Côté, and F. Maamari, Structural test with functional characteristics, in *Proc. IEEE Current and Defect-Based Testing Workshop*, pp. 57–60, May 2005.

[Nadeau-Dostie 2006] B. Nadeau-Dostie and J.-F. Côté, Clock Controller for At-Speed Testing of Scan Circuits U.S. Patent No. 7,155,651, December 26 2006.

[Naruse 2003] M. Naruse, I. Pomeranz, S. M. Reddy, and S. Kundu, On-chip compression of output responses with unknown values using LFSR reseeding, in *Proc. IEEE Int. Test Conf.*, pp. 1060–1068, October 2003.

[Pandey 2002] A. R. Pandey and J. H. Patel, Reconfiguration technique for reducing test time and test volume in Illinois scan architecture based designs, in *Proc. IEEE VLSI Test Symp.*, pp. 9–15, April 2002.

[Patel 2003] J. H. Patel, S. S. Lumetta, and S. M. Reddy, Application of Saluja-Karpovsky compactors to test responses with many unknowns, in *Proc. IEEE VLSI Test Symp.*, pp. 107–112, April 2003.

[Pouya 1998] B. Pouya and N. A. Touba, Synthesis of zero-aliasing space elementary-tree space compactors, in *Proc. IEEE VLSI Test Symp.*, pp. 70–77, April 1998.

[Rajski 2004] J. Rajski, J. Tyszer, M. Kassab, and N. Mukherjee, Embedded deterministic test, *IEEE Trans. on Computer-Aided Design*, 23(5), pp. 776–792, May 2004.

[Rajski 2005] J. Rajski, J. Tyszer, C. Wang, and S. M. Reddy, Finite memory test response compactors for embedded test applications, *IEEE Trans. on Computer-Aided Design*, 24(4), pp. 622–634, April 2005.

[Rajski 2008] J. Rajski, J. Tyszer, G. Mrugalski, W.-T. Cheng, N. Mukherjee, and M. Kassab, X-Press: Two-stage X-tolerant compactor with programmable selector, *IEEE Trans. on Computer-Aided Design*, 27(1), pp. 147–159, January 2008.

[Saluja 1983] K. K. Saluja and M. Karpovsky, Test compression hardware through data compression in space and time, in *Proc. IEEE Int. Test Conf.*, pp. 83–88, October 1983.

[Samaranayake 2003] S. Samaranayake, E. Gizdarski, N. Sitchinava, F. Neuveux, R. Kapur, and T. W. Williams, A reconfigurable shared scan-in architecture, in *Proc. IEEE VLSI Test Symp.*, pp. 9–14, April 2003.

[Shah 2004] M. A. Shah and J. H. Patel, Enhancement of the Illinois scan architecture for use with multiple scan inputs, in *Proc. IEEE Computer Society Annual Symp. on VLSI*, pp. 167–172, February 2004.

[Sitchinava 2004] N. Sitchinava, S. Samaranayake, R. Kapur, E. Gizdarski, F. Neuveux, and T. W. Williams, Changing the scan enable during shift, in *Proc. IEEE VLSI Test Symp.*, pp. 73–78, April 2004.

[Synopsys 2008] Synopsys, http://www.synopsys.com, 2008.

[SynTest 2008] SynTest Technologies, http://www.syntest.com, 2008.

[Touba 2006] N. A. Touba, Survey of test vector compression techniques, *IEEE Design & Test of Computers*, 23(4), pp. 294–303, July-August 2006.

[Touba 2007] N. A. Touba, X-canceling MISR—An X-tolerant methodology for compacting output responses with unknowns using a MISR, in *Proc. IEEE Int. Test Conf.*, Paper 6.2, October 2007.

[Touba 2008] N. A. Touba and L.-T. Wang, X-Canceling Multiple-Input Signature Register (MISR) for Compacting Output Responses with Unknowns, U.S. Patent Application No. 12,007,693, January 14, 2008.

[Volkerink 2005] E. H. Volkerink and S. Mitra, Response compaction with any number of unknowns using a new LFSR architecture, in *Proc. ACM/IEEE Design Automation Conf.*, pp. 117–122, June 2005.

[Wang 2002] L.-T. Wang, H.-P. Wang, X. Wen, M.-C. Lin, S.-H. Lin, D.-C. Yeh, S.-W. Tsai, K. S. Abdel-Hafez, Method and Apparatus for Broadcasting Scan Patterns in a Scan-Based Integrated Circuit, U.S. Patent Application No. 20030154433, January 16, 2002.

[Wang 2004] L.-T. Wang, X. Wen, H. Furukawa, F.-S. Hsu, S.-H. Lin, S.-W. Tsai, K. S. Abdel-Hafez, and S. Wu, VirtualScan: A new compressed scan technology for test cost reduction, in *Proc. IEEE Int. Test Conf.*, pp. 916–925, October 2004.

[Wang 2008] L.-T. Wang, X. Wen, S. Wu, Z. Wang, Z. Jiang, B. Sheu, and X. Gu, VirtualScan: Test compression technology using combinational logic and one-pass ATPG, *IEEE Design & Test of Computers*, 25(2), pp. 122–130, March-April 2008.

[Wohl 2001] P. Wohl, J. A. Waicukauski, and T. W. Williams, Design of compactors for signature-analyzers in built-in self-test, in *Proc. IEEE Int. Test Conf.*, pp. 54–63, October 2001.

[Wohl 2003] P. Wohl, J. A. Waicukauski, S. Patel, and M. B. Amin, Efficient compression and application of deterministic patterns in a logic BIST architecture, in *Proc. ACM/IEEE Design Automation Conf.*, pp. 566–569, June 2003.

[Wohl 2004] P. Wohl, J. A. Waicukauski, and S. Patel, Scalable selector architecture for X-tolerant deterministic BIST, in *Proc. ACM/IEEE Design Automation Conf.*, pp. 934–939, June 2004.

[Wohl 2007a] P. Wohl, J. A. Waicukauski, R. Kapur, S. Ramnath, E. Gizdarski, T. W. Williams, and P. Jaini, Minimizing the impact of scan compression, in *Proc. IEEE VLSI Test Symp.*, pp. 67–74, May 2007.

[Wohl 2007b] P. Wohl, J. A. Waicukauski, and S. Ramnath, Fully X-tolerant combinational scan compression, in *Proc. IEEE Int. Test Conf.*, Paper 6.1, October 2007.

R3.6 Concluding Remarks

[Bardell 1982] P. H. Bardell and W. H. McAnney, Self-testing of multiple logic modules, in *Proc. IEEE Int. Test Conf.*, pp. 200–204, November 1982.

[Naffziger 2006] S. Naffziger, B. Stackhouse, T. Grutkowski, D. Josephson, J. Desai, E. Alon, and M. Horowitz, The implementation of a 2-core multi-threaded Itanium family processor, *IEEE J. of Solid-State Circuits*, 41(1), pp. 197–209, January 2006.

[Stackhouse 2008] B. Stackhouse, B. Cherkauer, M. Gowan, P. Gronowski, and C. Lyles, A 65 nm 2-billion-transistor quad-core Itanium processor, *Digest of Papers, IEEE Int. Solid-State Circuits Conf.*, pp. 92, February 2008.

[Wang 1986b] L.-T. Wang and E. J. McCluskey, Concurrent built-in logic block observer (CBILBO), in *Proc. IEEE Int. Symp. on Circuits and Systems*, 3(3), pp. 1054–1057, May 1986.